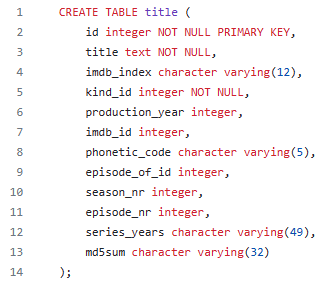
# 基于机器学习的数据库基数估计任务

## 任务描述

**基数估计**是数据库中一个重要的模块。对于输入的查询，基数估计模块将快速估计其满足不同执行顺序条件下的记录行数，其中记录行数也称为基数。通过对不同执行顺序的基数估计，可以选择出最优的执行顺序，也称为执行计划。本任务希望利用机器学习模型实现准确的基数估计。

下面通过具体例子介绍本任务。如下是IMDB互联网电影资料库中一张名为title的数据表的模式（Schema）：



例如该表包含 production\_year (发行的年份)，kind\_id (电影种类类型)，imdb\_id (这部电影在IMDB中的id号)等描述电影的字段，这些字段在数据表内以列的形式展现。以下是一些数据查询的例子：

1） 如果想查询发行的年份大于1970年的所有电影，可使用如下的sql语句：

**select \* from title where production\_year > 1970;**

2） 如果想查询发行的年份小于或等于2010年的所有电影，可使用如下的sql语句：

**select \* from title where production\_year <= 2010;**

3） 如果想查询发行的年份大于1970年同时发行的年份小于等于2010年的所有电影，可使用如下的sql语句：

**select \* from title where production\_year > 1970 and production\_year <= 2010;**

4） 如果想查询在IMDB数据库中id小于20000同时发行的年份大于1988年的所有电影，可使用如下的sql语句：

**select \* from title where imdb\_id < 20000 and production\_year > 1988;**

以查询4为例，在执行**select \* from title where imdb\_id < 20000 and production\_year > 1988;** 查询语句时存在一个执行顺序的问题，是先查询发行年份大于1988年的所有电影，再在其中查询imdb\_id小于20000的基数更少，还是先查询imdb\_id小于20000的所有电影，再在其中查询发行年份大于1988年的电影的基数更少呢？不难发现，满足查询条件的最终基数是相同的，但达到最终结果的过程中，不同的执行计划引入了不同的基数，而这便是基数估计存在的意义，决定最优的（查询过程中基数的和是最小的）执行顺序。

结合在关系数据库中诸如 production\_year, kind\_id, imdb\_id 等字段在数据表内以列的形式展现。我们可以采用column1代替production\_year, column2代替kind\_id, column3代替imdb\_id。那么查询便可以统一抽象为下述sql查询语句表示：

**select \* from title where column1 > 1970 and column1 < 2010;**

**在本任务中，进一步约定各个字段属性皆为integer整数类型**。因此，涉及到的查询 production\_year > 1970 and production\_year < 2010 便可抽象的表示为 column1 > 1970 and column1 < 2010, 其中我们把 1970 称作column1 的下界，用lower\_column1表示，将2010称作column1的上界，用upper\_column1表示，为了书写方便，进一步用 c1L表示lower\_column1, 用c1R表示upper\_column1, 那么对于给定的查询，便可用统一的查询表达式进行表示，表达式如下所示：

**select \* from title where column1 > lower\_column1 and column1 < upper\_column1 and column2 > lower\_column2 and column2 < upper\_column2 and … and columnN > lower\_columnN and columnN < upper\_columnN;**

对于未给定的范围，例如查询发行年份大于1988年的所有电影，可以自然的将查询语句改写为查询发现年份大于1988年同时发行年份小于正无穷大 (+∞) 的所有电影。基于抽象，我们便可用如下统一范式对查询语句进行建模：

[c1L, c1R, c2L, c2R, …, cNL, cNR]

其中ciL的默认值是负无穷大 (-∞)，ciR的默认值为正无穷大 (+∞)。对于第i列已经给定查询范围，其对应的ciL为定值或ciR是定值或是两个值均为定值。为了方便起见，本任务给出的查询仅针对类型为integer的类，便可统一用此范式表示。

将上述特征输入机器学习模型，即可预测该特征对应的查询结果的基数 est\_rows；我们期望模型估计的基数 est\_rows 与实际查询执行的基数 act\_rows (在数据集中给出，基于真实的数据库得到)之间的差距越小越好。

## 补充知识

上述建模过程中，使用的特征较为简单，在数据量不是很充分的情况下，你的模型可能不能很好的学习到查询语句与基数之间的映射关系。这里介绍3个启发式方法得到的指标，从而让模型利用这些指标更好地进行预测。更多的特征提取方法见文献[1-4]:

* 1. Attribute Value Independence (**AVI**): 此方法认为所有查询条件的选择率 (selectivity) 满足乘法原则，选择率代表某个查询查询出的记录行数占总行数的百分比，例如假设某数据集中2010年后（包括2010年）生产的电影占数据集中总数的25%，那么对应查询语句 production\_year >= 2010 and production\_year <= +∞ 对应的选择率便为 25%。回到 **AVI** 方法本身，其认为所有查询条件的选择率满足乘法原则，例如假设某数据集中2010年后（包括2010年）生产的电影占数据集中总数的25%，而该数据集中爱情电影所占的比例是10%，根据**AVI**方法，易得出2010年后（包括2010年）生产的**爱情电影**占总数据集的2.5% (2.5% = 25% \* 10%)。那么**AVI**方法便得出其总的选择率为 s1 X s2 X s3 X … X sN其中si 是第i-th column的选择率，通过将总的选择率乘上数据总行数，便得出**AVI**方法估计出来的基数。
  2. Exponential BackOff (**EBO**):此方法不同**AVI**方法考虑所有的列，其认为应选出4个最小的选择率组成总的选择率，结合得到如下的式子:

,

其中代表第k小的选择率。通过将总的选择率乘上数据总行数，便得出**EBO**方法估计出来的基数。

* 1. Minimum Selectivity (**MinSel**):该方法较为简单，其认为总的选择率便是最小的选择率。通过将总的选择率乘上数据总行数，便得出**MinSel**方法估计出来的基数。

下面给出**AVI**方法的代码实现，本任务要求你以AVI方法的代码实现为参考，自己实现**EBO**方法与**MinSel**方法。

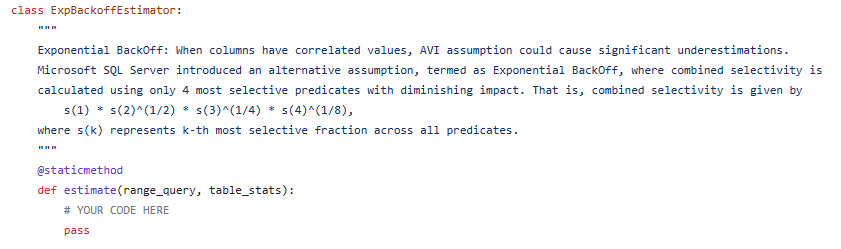
图形用户界面, 文本, 应用程序

描述已自动生成

代码中的table\_stats将在 **3.数据集说明** 中介绍。在IMDB数据集中，其中一个 column\_name 便是production\_year, 其属于 range\_query.column\_names() 中。以production\_year 为例，对于如下sql:

select \* from title where production\_year > 1988 and production\_yaer < 2010;

首先会得到 min\_val = 1988 同时 max\_val = 2010，对于一些未出现的column或是column中仅有上界或仅有下界，符合min\_val 默认值为负无穷大 (-∞)，max\_val 默认值为正无穷大 (﹢∞)。通过min\_val 与 max\_val 查找其在table\_stats内的对应的块的id: left 与 right (table\_stats本质是通过直方图对数据分布进行分块) ，通过left，right可以得到此查询实际访问的行数 col\_cnt, 注意col\_cnt 的过程中将 left, right 变为了 left+1, right 主要是考虑了区间左闭右开，原始查询为左开右开，因而加了1。基于此，便可得到production\_year对应的selectivity，根据**AVI**方法的设计原则，其采用一个变量**sel**不断累乘这些selectivity，变得到了总的选择率。



**EBO**方法如上图所示，需要你进行完善。

图形用户界面

中度可信度描述已自动生成

**MinSel**方法如上图所示，需要你进行完善。

通过实现了**EBO**方法与**MinSel**方法。你可将之前的特征拓展如下：

[c1L, c1R, c2L, c2R, …, cNL, cNR, est\_avi, est\_ebo, est\_min\_sel]

其中 est\_avi 为通过**AVI**方法得到的总的选择率，est\_ebo为通过**EBO**方法得到的总的选择率，est\_min\_sel为通过**min\_sel**方法得到的总的选择率，你可使用这些指标来辅助你的模型得到更准确的总的选择率，总的选择率乘上数据表总行数便为基数。

你可以通过阅读 Selectivity Estimation for Range Predicates using Lightweight Models[5] 获得更多细节。

## 数据集说明

**title\_stats.json**

title\_stats.json为使用直方图histogram方法对IMDB数据集也就是本任务中的title表进行分析得到的json文件，因此命名为title\_stats.json。以下图为例，其为json文件，columns包含的第一个字段为 episode\_nr，



title\_stats.json 对 title 表所有字段都采用了直方图进行统计。以episode\_nr 为例，

图形用户界面, 应用程序

描述已自动生成

其对应title表中的episode\_nr字段，可以看到的采用的是 histogram 方法，其将分布划分为buckets，通过遍历buckets即可快速知道查询命中的行数。例如min\_val 和 max\_val 可通过第一个bucket与最后一个bucket得到。

图形用户界面

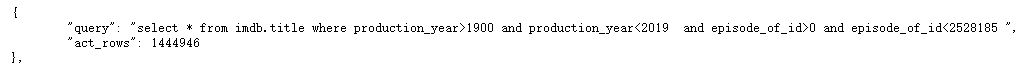
中度可信度描述已自动生成

**query\_train\_18000.json**

文本

中度可信度描述已自动生成

训练数据如图所示，包含18000条训练数据，可以通过下图看到训练数据的关键是给出了执行查询语句与其对应的真实基数：



**validation\_2000.json**

2000条测试数据，作为你模型的验证集。

此外，可以根据需要自行划分训练集和验证集，提高模型泛化性。

## 评测指标

本任务采用查询的估计执行基数 est\_rows 与实际查询执行的真实基数 act\_rows (由数据集提供) 之间的差距进行评测。具体计算公式如下:

Loss越低意味着模型对于测试数据基数估计的越准。

## 参考文献

[1] How the planner uses statistics. https://www.postgresql.org/docs/current/rowestimation-examples.html.

[2] Statistics in Microsoft SQL Server 2017. https://docs.microsoft.com/enus/sql/relational-databases/statistics/statistics?view=sql-server-2017.

[3] Understanding optimizer statistics with oracle database 18c. https://www.oracle.com/technetwork/database/bi-datawarehousing/twpstats-concepts-0218-4403739.pdf.

[4] N. Bruno, S. Chaudhuri, and L. Gravano. Stholes: A multidimensional workload-aware histogram. In SIGMOD, 2001.

[5] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek R. Narasayya, Surajit Chaudhuri: Selectivity Estimation for Range Predicates using Lightweight Models. Proc. VLDB Endow. 12(9): 1044-1057 (2019)