查询规模估算
Cardinality Estimation

2025-5-25

查询规模定义

- 查询规模 (Cardinality)

  Q: SELECT *
  FROM $Student$ WHERE $age > 15$
  AND $gender$ = 'Male';

  $\mathrm{Card(Q)} = 4$

- 实验目标：给定一条SQL查询，

  预测符合该查询的结果数目

age	gender	GPA
21	Female	3.42
20	Male	2.58
18	Female	2.79
20	Female	3.98
24	Female	3.71
20	Male	3.50
21	Male	4.0
23	Female	3.66
22	Male	3.12

Why Cardinality Estimation

2014

IS QUERY OPTIMIZATION A "SOLVED" PROBLEM?

Databases

Guy Lohman, IBM DB2 (40 years' experience)

"The root of all evil, the Achilles Heel of query optimization, is the estimation of the size of intermediate results, known as cardinalities."

2015

How Good Are Query Optimizers, Really?

"We have also shown that relational database systems produce large estimation errors that quickly grow as the number of joins increases, and that these errors are usually the reason for bad plans."

2018 - 2024

Multiple research groups consistently working on learned selectivity estimators

Massachusetts Institute of Technology

Berkeley UNIVERSITY OF CALIFORNIA

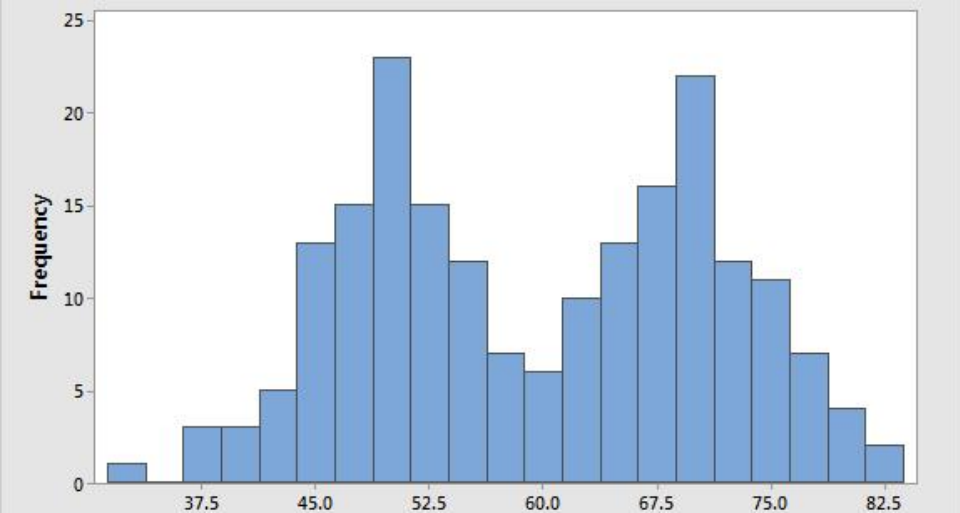
Technical University of Munich

Microsoft

Alibaba Group

传统规模估算方法

- Histograms
- Sampling



When Traditional Methods Fail

```
SELECT * FROM title   WHERE production_year > 1995
SELECT * FROM title   WHERE production_year > 2010
SELECT * FROM title   WHERE production_year > 2018
```

非均匀分布

```
SELECT * FROM title AS t
        WHERE t.production_year > 2005 AND revenues >100,000,000
```

属性相关

```
SELECT * FROM title AS t, movie_info_idx AS mi_idx
        WHERE t.production_year > 2005 AND   mi_idx.info < 8.5
        AND t.id = mi_idx.movie_id
```

JOIN

How Learned Cardinality Estimators Work

Methodology 1: **Query-driven**

- Key Idea: Model as a Regression problem

Query → (Feature Engineering) → Feature Vector → (ML Models) → Selectivity

Methodology 2: **Data-driven**

- Key Idea**:** Model as a Joint Distribution Estimation problem

A_1	A_2	...	A_n

$P(A_1, A_1, \dots, A_n)$

Methodology: Query-Driven

Training

Query Pool

Labels

$Q1$: SELECT *FROM $Student$ WHERE age > 20; 4
$Q2$: SELECT *FROM $Student$ WHERE GPA < 3.5 AND GPA > 3.0; 2
$Q3$: SELECT *FROM $Student$ WHERE $gender$ = 'Female'; 5
... ...

Featurize

$Q1$: <0.8, 1.0, 0.0, 0.0, 0.0, 1.0> 4
$Q2$: <0.0, 1.0, 0.0, 1.0, 0.3, 0.6> 2
$Q3$: <0.0, 1.0, 1.0, 1.0, 0.0, 1.0> 5
...

Train

Regression Model

Inference

Q: SELECT * FROM $Student$
WHERE age > 15 AND gender = "Male"

Featurize

Q: <0.0, 0.9, 0.0, 1.0, 0.8, 1.0>

Inference

Estimation: 4!

任务说明

根据SQL查询语句（计划），预测查询规模

SQL类型

- 数值型数据（范围查询与等值查询）
- 最多涉及两表连接
- 提供查询计划（可选）

提供60000条训练集，需要预测1070条测试集的结果

数据解释


```
[
    {
        "query": "SELECT * FROM movie_companies mc,title t,movie_info_idx mi_idx WHERE t.id=mc.movie_id AND t.id=mi_idx.movie_id AND mi_idx.info_type_id=112 AND mc.company
        "query_id": 0,
        "explain_result": "{\"QUERY PLAN\": [{\"Plan\": {\"Node Type\": \"Nested Loop\", \"Parallel Aware\": false, \"Async Capable\": false, \"Join Type\": \"Inner\", \"S
    },
    {
        "query": "SELECT * FROM movie_companies mc,title t,movie_info_idx mi_idx WHERE t.id=mc.movie_id AND t.id=mi_idx.movie_id AND mi_idx.info_type_id=113 AND mc.company
        "query_id": 1,
        "explain_result": "{\"QUERY PLAN\": [{\"Plan\": {\"Node Type\": \"Nested Loop\", \"Parallel Aware\": false, \"Async Capable\": false, \"Join Type\": \"Inner\", \"S
    },
    {
        "query": "SELECT * FROM movie_companies mc,title t,movie_info_idx mi_idx WHERE t.id=mc.movie_id AND t.id=mi_idx.movie_id AND mi_idx.info_type_id=112 AND mc.company
        "query_id": 2,
        "explain_result": "{\"QUERY PLAN\": [{\"Plan\": {\"Node Type\": \"Nested Loop\", \"Parallel Aware\": false, \"Async Capable\": false, \"Join Type\": \"Inner\", \"S
    },
    {
```


SQL查询
查询ID
查询计划

数据解释


```
{
    "QUERY PLAN": [
        {
            "Plan": {
                "Node Type": "Gather",
                "Parallel Aware": false,
                "Async Capable": false,
                "Startup Cost": 49254.42,
                "Total Cost": 97582.87,
                "Plan Rows": 321675,
                "Plan Width": 20,
                "Actual Startup Time": 178.037,
                "Actual Total Time": 307.563,
                "Actual Rows": 564144,
                "Actual Loops": 1,
                "Workers Planned": 2,
                "Workers Launched": 2,
                "Single Copy": false,
                "Plans": [
                    {
                        "Node Type": "Hash Join",
                        "Parent Relationship": "Outer",
                        "Parallel Aware": true,
                        "Async Capable": false,
                        "Join Type": "Inner",
                        "Startup Cost": 48254.42,
                        "Total Cost": 64415.37,
                        "Plan Rows": 134031,
                        "Plan Width": 20,
                        "Actual Startup Time": 172.353,
                        "Actual Total Time": 272.969,
                        "Actual Rows": 188048,
                        "Actual Loops": 3,
                        "Inner Unique": true,
                        "Hash Cond": "(mi_idx.movie_id = t.id)",
                        "Workers": [],
                        "Plans": [
                            {
                                "Node Type": "Seq Scan",
                                "Parent Relationship": "Outer",
                                "Parallel Aware": true,
                                "Async Capable": false,
                                "Relation Name": "movie_info_idx",
                                "Alias": "mi_idx",
                                "Startup Cost": 0.0,
                                "Total Cost": 15155.68,
                                "Plan Rows": 382960,
                                "Plan Width": 8,
                                "Actual Startup Time": 0.017,
                                "Actual Total Time": 42.625,
                                "Actual Rows": 306703,
                                "Actual Loops": 3,
                                "Filter": "(info_type_id > 99)",
                                "Rows Removed by Filter": 153308,
                                "Workers": []
                            },
```


如何load plan

```
def load_plan():
    explain_result_str = item['explain_result']
    plan = json.loads(explain_result_str)
```

Column_min_max_vals.csv

```
name,min,max,cardinality,num_unique_values
t.id,1,2528312,2528312,2528312
t.kind_id,1,7,2528312,6
t.production_year,1880,2019,2528312,133
mc.id,1,2609129,2609129,2609129
mc.company_id,1,234997,2609129,234997
mc.movie_id,2,2525745,2609129,1087236
mc.company_type_id,1,2,2609129,2
ci.id,1,36244344,36244344,36244344
ci.movie_id,1,2525975,36244344,2331601
ci.person_id,1,4061926,36244344,4051810
ci.role_id,1,11,36244344,11
mi.id,1,14835720,14835720,14835720
mi.movie_id,1,2526430,14835720,2468825
mi.info_type_id,1,110,14835720,71
mi_idx.id,1,1380035,1380035,1380035
mi_idx.movie_id,2,2525793,1380035,459925
mi_idx.info_type_id,99,113,1380035,5
mk.id,1,4523930,4523930,4523930
mk.movie_id,2,2525971,4523930,476794
mk.keyword_id,1,134170,4523930,134170
```

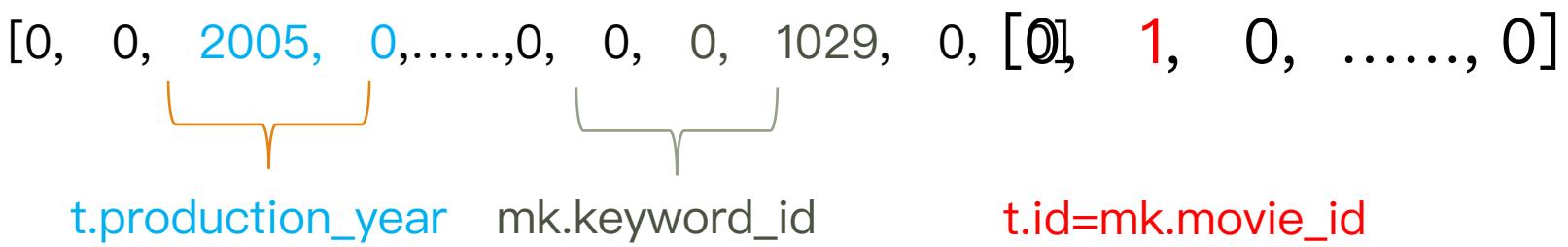
每张表的最小值、最大值、元组数目、不重复记录数目

Submission sample

```
Query ID,Predicted Cardinality
0,0
1,0
2,2
3,6
4,12
5,20
```

A naïve implementation

- 共有N个表，d个属性
- lb: lower bound　ub: upper bound
- 2*d维向量 [lb_1,ub_1,lb_2,ub_2,…….lb_d,ub_d]
- One-hot encoding for join
- SELECT * FROM title t, movie_keyword mk WHERE t.production_year >2005 AND mk.keyword_id<1029 AND t.id=mk.movie_id



Representative Works

Query-based

- **MSCN** [Kipf, A et all. CIDR 19]

  Neural Network + Sampling

- **LW-XGB** [Dutt, A et all. VLDB 19]

  XGBoost+ Histogram

- **LW-NN** [Dutt, A et all. VLDB 19]

  Neural Network + Histogram

- **QuickSel** [Yongjoo, P et all. SIGMOD 20]

  Mixture Model

Data-based

- **Naru** [Yang, Z et all. VLDB 20]

  Auto-regressive Model

- **DeepDB** [Hilprecht, B et all. VLDB 20]

  Sum Product Network

- **FLAT** [Rong, Z et all. VLDB 2021]

  Graphical Model

Plan

- QueryFormer(VLDB 2022)
  - Transformer
- NEO(VLDB 2019)
  - Tree-CNN
- E2E-COST(VLDB 2019)
  - Tree-LSTM