SQL for SRL: Structure Learning Inside a Database System

Oliver Schulte and Zhensong Qian, Simon Fraser University, Canada

Introduction

- **Statistical-Relational Learning**: Learn a joint statistical model for *all* tables in the input database.
- New approach to *SRL system building*:
- The RDBMS stores structured objects for statistical analysis as *first-class citizens* in the database.
- *SQL* is used to build and transform statistical objects:
  - Structured Model (*Bayesian network, Markov Logic Network*).
  - Parameter Estimates.
  - Sufficient Statistics.
- Empirical evaluation: leveraging the RDBMS capabilities achieves scalable learning and fast model testing.
- All code and datasets are available online [1].

Contributions

- Identifying new system requirements for multi-relational machine learning that go beyond single table machine learning.
- An integrated set of SQL-based solutions for providing these system capabilities.

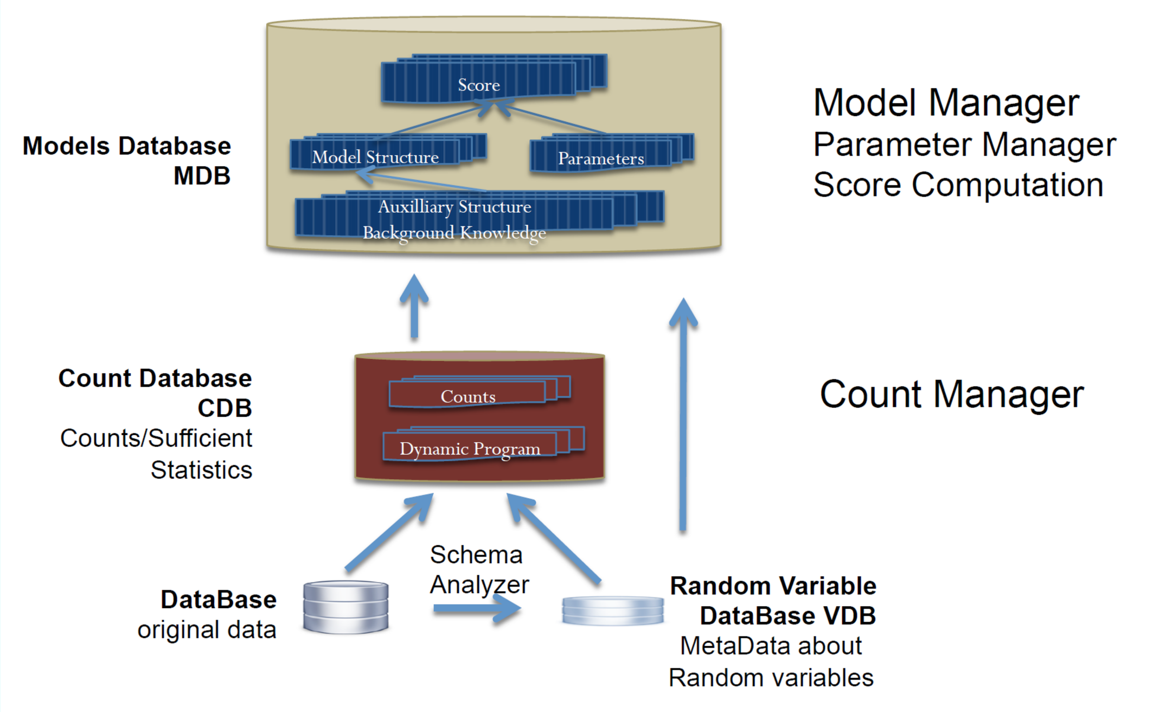
Related Works

- BayesStore [3]: all statistical objects are first-class citizens in a relational database. Inference, no learning.
- MadLib [5]: leverages SQL for *single-relational* data table analysis.
- Tuffy [7]: reliable and scalable inference and parameter learning for Markov Logic Networks with an RDBMS. No structure learning.

References

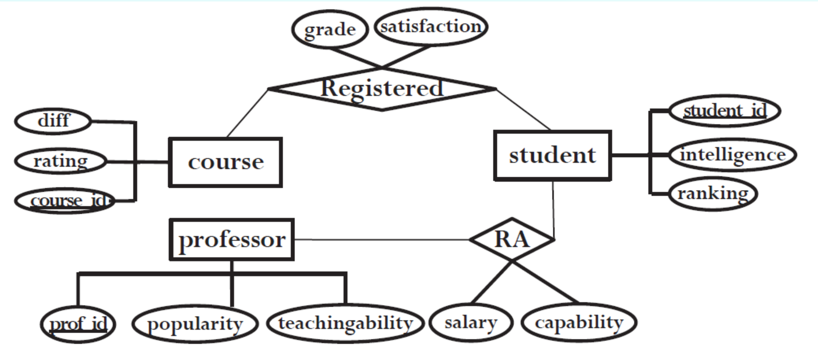
1. Qian, Z.; Schulte, O. The BayesBase System. www.cs.sfu.ca/~oschulte/BayesBase
2. Russell, S. & Norvig, P. Artificial Intelligence: A Modern Approach Prentice Hall, 2010.
3. Wang, D. Z.; Michelakis, E.; & et al. BayesStore: managing large, uncertain data repositories with probabilistic graphical models, PVLDB, 2008, 1, 340-351.
4. Qian, Z.; Schulte, O. & Sun, Y. Computing Multi-Relational Sufficient Statistics for Large Databases, CIKM 2014, 1249-1258.
5. Hellerstein, J. M.; Ré, C.; Schoppmann, F.; & et al, The MADlib Analytics Library: Or MAD Skills, the SQL, PVLDB, 2012, 5, 1700-1711
6. Schulte, O. & Khosravi, H. Learning graphical models for relational data via lattice search Machine Learning, 2012, 88, 331-368
7. Niu, F.; Ré, C.; Doan, A. & Shavlik, J. W. Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS PVLDB, 2011, 4, 373-384

System Overview



- **Schema Analyzer:** examines the information in the DB system catalog to define a default set of random variables.
- **Count Manager:** uses the meta data in the VDB database to compute multi-relational sufficient statistics for a set of random variables [4].
- **Model Manager:** supports the construction and querying of large structured statistical models.

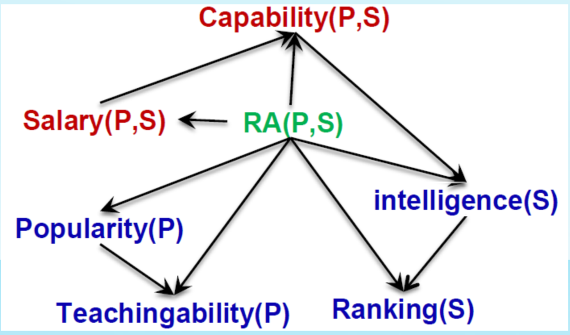
ER-Design for University Domain



The Model Manager

Goal: Learn First-Order Bayesian Network [2].

- Bayesian Network Structure Learning [6].
- Nodes = Random Variables
- Edges are stored in Database tables
- Model selection scores are also stored, not shown (BIC, AIC, BDeu)



Child	Parent
Capability(P,S)	RA(P,S)
Capability(P,S)	Salary(P,S)
Teachingability(P)	Popularity(P)
Teachingability(P)	RA(P,S)
...	...

The Parameter Manager

Goal: Learn Bayesian Network Parameters

- Stored in Conditional Probability (CP) table.
- Maximum Likelihood Estimate are easy to compute from database counts.

Capa(P,S)	RA(P,S)	Salary(P,S)	CP
4	T	high	0.45
5	T	high	0.36
3	T	high	0.18
3	T	low	0.20
2	T	low	0.40
1	T	low	0.40
2	T	med	0.22
3	T	med	0.44
1	T	med	0.33

CP table

Count	Capa(P,S)	RA(P,S)	Salary(P,S)
5	4	T	high
4	5	T	high
2	3	T	high
1	3	T	low
2	2	T	low
2	1	T	low
2	2	T	med
4	3	T	med
3	1	T	med

Contingency Table

```
SELECT COUNT(*) AS Count,
Capability as `Capa(P,S)`, 'T'
as `RA(P,S)`, Salary as
`Salary(P,S)`
FROM `RA`;
```

Specific SQL Query

The Random Variable Database

<table>
<tr><th>Table Name</th><th colspan="6">Column Headers in Random Variable Database</th></tr>
<tr><td rowspan="4">Pvariables</td><td>Pvid</td><td colspan="5">TABLE_NAME</td></tr>
<tr><td>C</td><td colspan="5">course</td></tr>
<tr><td>P</td><td colspan="5">prof</td></tr>
<tr><td>S</td><td colspan="5">student</td></tr>
<tr><td rowspan="7">1Variables</td><td>1VarID</td><td colspan="3">COLUMN_NAME</td><td colspan="2">Pvid</td></tr>
<tr><td>diff(C)</td><td colspan="3">diff</td><td colspan="2">C</td></tr>
<tr><td>intelligence(S)</td><td colspan="3">intelligence</td><td colspan="2">S</td></tr>
<tr><td>popularity(P)</td><td colspan="3">popularity</td><td colspan="2">P</td></tr>
<tr><td>ranking(S)</td><td colspan="3">ranking</td><td colspan="2">S</td></tr>
<tr><td>rating(C)</td><td colspan="3">rating</td><td colspan="2">C</td></tr>
<tr><td>teachingability(P)</td><td colspan="3">teachingability</td><td colspan="2">P</td></tr>
<tr><td rowspan="5">2Variables</td><td>2VarID</td><td>COLUMN_NAME1</td><td colspan="2">COLUMN_NAME2</td><td>Pvid1</td><td>Pvid2</td></tr>
<tr><td>capability(P,S)</td><td>p_id</td><td colspan="2">s_id</td><td>P</td><td>S</td></tr>
<tr><td>grade(C,S)</td><td>c_id</td><td colspan="2">s_id</td><td>C</td><td>S</td></tr>
<tr><td>salary(P,S)</td><td>p_id</td><td colspan="2">s_id</td><td>P</td><td>S</td></tr>
<tr><td>sat(C,S)</td><td>c_id</td><td colspan="2">s_id</td><td>C</td><td>S</td></tr>
<tr><td rowspan="3">Relationship</td><td>RVarID</td><td>TABLE_NAME</td><td>COLUMN_NAME1</td><td>COLUMN_NAME2</td><td>Pvid1</td><td>Pvid2</td></tr>
<tr><td>RA(P,S)</td><td>RA</td><td>p_id</td><td>s_id</td><td>P</td><td>S</td></tr>
<tr><td>Registered(C,S)</td><td>Registered</td><td>c_id</td><td>s_id</td><td>C</td><td>S</td></tr>
</table>

Meta data about random variables stored in database tables.

- Domain of possible values.
- Pointer to corresponding data table/column.
- ...

The Count Manager

Goal: for a conjunctive query, compute the instantiation count = result set size.

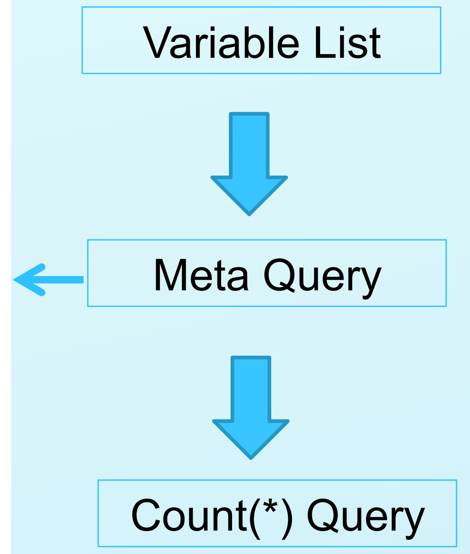
- Stored in Contingency (CT) Table [4].
- Main computational cost in learning.

Problem: need to generate SQL queries for **arbitrary variable lists**.
Solution: use Meta Data + **Meta Queries**

General Form of SQL Count Query:

```
SELECT COUNT(*) AS Count, <VARIABLE-LIST>
FROM TABLE-LIST
GROUP BY <VARIABLE-LIST>
WHERE <Join-Conditions>
```

<table>
<tr><th>Metaqueries</th><th>Entries</th></tr>
<tr><td rowspan="5">CREATE TABLE Select_List AS
SELECT RVarID, CONCAT('COUNT(*)',' as "count"') AS Entries
FROM Relationship
UNION DISTINCT
SELECT RVarID, 1VarID AS Entries
FROM Relationship_1Variables;</td><td>COUNT(*) as "count"</td></tr>
<tr><td>`popularity(P)`</td></tr>
<tr><td>`teachingability(P)`</td></tr>
<tr><td>`intelligence(S)`</td></tr>
<tr><td>`ranking(S)`</td></tr>
<tr><td rowspan="3">CREATE TABLE From_List AS
SELECT RVarID, CONCAT('@database@.',TABLE_NAME) AS Entries
FROM Relationship_Pvariables
UNION DISTINCT
SELECT RVarID, CONCAT('@database@.',TABLE_NAME) AS Entries
FROM Relationship;</td><td>@database@.prof AS P</td></tr>
<tr><td>@database@.student AS S</td></tr>
<tr><td>@database@.RA AS `RA`</td></tr>
<tr><td rowspan="2">CREATE TABLE Where_List AS
SELECT
RVarID, CONCAT(RVarID,'.',COLUMN_NAME,' = ',
Pvid,'.', REFERENCED_COLUMN_NAME) AS Entries
FROM Relationship_Pvariables;</td><td>`RA`.p_id = P.p_id</td></tr>
<tr><td>`RA`.s_id =S.s_id</td></tr>
</table>



Results

Task: learning a multi-relational Bayesian network

Dataset	# Database Tuples	# Sufficient Statistics (SS)	SS Computing Time (s)	#BN Parameters
Movielens	1,010,051	252	2.7	292
Mutagenesis	14,540	1,631	1.67	721
UW-CSE	712	2,828	3.84	241
Mondial	870	1,746,870	1,112.84	339
Hepatitis	12,927	12,374,892	3,536.76	569
IMDB	1,354,134	15,538,430	7,467.85	60,059

Database and performance statistics for MRLBase

Comparison with other statistical-relational learning (Markov Logic Networks)

Dataset	RDN_Boost	MLN_Boost	MRLBase	MRLBase-CT
MovieLens	92.7min	N/T	1.12	0.39
Mutagenesis	118	49	1	0.15
UW-CSE	15	19	1	0.27
Mondial	27	42	102	61.82
Hepatitis	251	230	286	186.15
IMDB	N/T	N/T	524.25	439.29

The RDBMS support for multi-relational learning translates into orders of magnitude improvements in speed and scalability.

Speedup on other tasks: compute model selection score, test models, cross-validation. Not shown.

Conclusions

- Multi-relational learning requires new system capabilities.
  - ➢ leverage SQL, RDBMS.
- Fast system development through high-level SQL constructs.
- Manage large statistical objects: parameters, sufficient statistics.
- Fast native support for counting (count(*)).
- Future Directions:
  - ➢ distributed processing, in-memory computing (SparkSQL)
  - ➢ Integrate with inference systems (BayesStore, Tuffy)