CS848 - Paper Critique

RDFBroker

“A Signature-Based High-Performance RDF Store”

by

Michael Sintek and Malte Kiesel

Presented by:
Atif Khan (Sep 24th, 2013)

CS848 Class Presentation
Motivation

RDF Triple Representation is Problematic

- *triple* knowledge representation leads to *system level inefficiencies*
Motivation

RDF Triple Representation is Problematic

– triple knowledge representation leads to system level inefficiencies

Inefficiency Types

run-time: not suited for application consumption

storage: inefficient handling in databases
Motivation

RDF Triple Representation is Problematic

- *triple* knowledge representation leads to system level inefficiencies

Inefficiency Types

**run-time:** not suited for application consumption (applications utilize object oriented model)

**storage:** inefficient handling in databases (duplication, joins)
Objective – Main Idea

Map RDF *triples* into database *tables*
Objective – Main Idea

Map RDF triples into database tables

{<s1> <p1> <>
 <s1> <p2> <>
 <s2> <p1> <>
 <s2> <p2> <>
 <s2> <p3> <>
 ... } ->

Sep 24, 2013
Objective – Main Idea

Map RDF *triples* into database *tables*
Objective – Main Idea

Map RDF *triples* into database *tables*

Considerations

- natural & schema independent mapping

- tables are constructed based on *properties* defined for *subjects*
Basic Concepts

Property Signature – (def 1)

\[ \sum_G (s) = \{ p \mid \exists o : \langle s, p, o \rangle \in G \} \]

a signature of a resource \( s \) is the set of properties w.r.t. a graph \( G \)
Basic Concepts

Property Signature – (def 1)

\[ \sum_{G}(s) = \{ p \mid \exists o : \langle s, p, o \rangle \in G \} \]

 RDF Broker
Basic Concepts

Property Signature – (def 1)

\[ \sum_G(s) = \{ p \mid \exists o : \langle s, p, o \rangle \in G \} \]

\[ \sum_G(Person) = \{ rdf : type \} \]
Basic Concepts

Property Signature – (def 1)

\[ \Sigma_G(s) = \{ p \mid \exists o : \langle s, p, o \rangle \in G \} \]

\[ \Sigma_G(Person) = \{ rdf : type \} \]

\[ \Sigma_G(p_3) = \{ rdf : type, fname, lname \} \]
Basic Concepts

Property Signature – *(def 1)*

\[
\sum_G(s) = \{ p \mid \exists o : \langle s, p, o \rangle \in G \}
\]

\[
\sum_G(Person) = \{ \text{rdf : type} \}
\]

\[
\sum_G(p_1) = \{ \text{rdf : type, label, fname, lname} \}
\]
Basic Concepts

Graph Signature Set – (def 2)

\[ \Sigma_G = \{ \sum_G(s) \mid \exists p, o : \langle s, p, o \rangle \in G \} \]

the graph signature set of a graph $G$ is the set of all property signatures in the graph $G$
Basic Concepts

Graph Signature Set – (def 2)

\[ \Sigma_G = \{ \sum_G(s) | \exists p, o : \langle s, p, o \rangle \in G \} \]

\[ \Sigma_G(Person) = \{ \text{rdf: type} \} , \]
\[ \Sigma_G(p_3) = \{ \text{rdf: type, fname, lname} \} , \]
\[ \Sigma_G(p_1) = \{ \text{rdf: type, label, fname, lname} \} \]
Basic Concepts

Property Signature Subsumtion ($\subseteq$) – (def 3)

$$\sum_G (Person) = \{rdf : type\}$$

$$\sum_G (p_3) = \{rdf : type, fname, lname\}$$

$$\sum_G (p_1) = \{rdf : type, fname, lname, label\}$$
Basic Concepts

Property Signature Subsumption ($\subseteq$) – (def 3)

\[
\sum_G(Person) = \{ \text{rdf : type} \}
\]

\[
\sum_G(Person) \subseteq \sum_G(p_3)
\]

\[
\sum_G(p_3) = \{ \text{rdf : type, fname, lname} \}
\]

\[
\sum_G(p_1) = \{ \text{rdf : type, fname, lname, label} \}
\]
Basic Concepts

Property Signature Subsumption ($\subseteq$) – (def 3)

\[ \sum_G (\text{Person}) = \{ \text{rdf : type} \} \]

\[ \sum_G (\text{Person}) \subseteq \sum_G (p_3) \]

\[ \sum_G (p_3) = \{ \text{rdf : type, fname, lname} \} \]

\[ \sum_G (p_3) \subseteq \sum_G (p_1) \]

\[ \sum_G (p_1) = \{ \text{rdf : type, fname, lname, label} \} \]
Basic Concepts

Signature Subsumtion Graph
Basic Concepts

Signature Subsumption Graph

\[ \Sigma_G \left( Person \right) = \{ rdf : type \} \]

\[ \Sigma_G \left( Person \right) \subseteq \Sigma_G \left( p_3 \right) \]

\[ \Sigma_G \left( p_3 \right) = \{ rdf : type, fname, lname \} \]

\[ \Sigma_G \left( p_3 \right) \subseteq \Sigma_G \left( p_1 \right) \]

\[ \Sigma_G \left( p_1 \right) = \{ rdf : type, fname, lname, label \} \]
Basic Concepts

Signature Subsumption Graph

\[
\{ \text{rdf : type} \}\\\
\sum_G (\text{Person}) \subseteq \sum_G (p_3)
\]

\[
\{ \text{rdf : type, fname, lname} \}\\\n\sum_G (p_3) \subseteq \sum_G (p_1)
\]

\[
\{ \text{rdf : type, label, fname, lname} \}
\]
Signature Subsumption Graph $\mathcal{G} = (\sum_G, \subseteq) - (def\ 4)$

- $\{\text{rdf:type}\}$
- $\{\text{rdf:type, fname, lname}\}$
- $\{\text{rdf:type, label, fname, lname}\}$
Basic Concepts

Signature Subsumption Graph \( \mathcal{G} = (\sum_G, \subseteq) - (\text{def } 4) \)

Properties

- directed acyclic graph
- vertices from graph signature set
- edges based on subsumption relationship

\[
\begin{align*}
\{ \text{rdf : type} \} & \quad \text{subsumes} \\
\{ \text{rdf : type, fname, lname} \} & \quad \text{subsumes} \\
\{ \text{rdf : type, label, fname, lname} \} &
\end{align*}
\]
Basic Concepts

Signature Subsumption Graph \( G = (\sum_G, \subseteq) - (def \ 4) \)

Properties

- directed acyclic graph
- vertices from \textit{graph signature set}
- edges based on \textit{subsumtion relationship}
Basic Concepts

Signature Subsumtion Graph $\mathcal{G} = (\sum_G, \subseteq) - (def\ 4)$

Properties

- directed acyclic graph
- vertices from graph signature set
- edges based on subsumtion relationship

\[
\begin{align*}
\{\text{rdf : type}\} & \quad \text{subsumes} \\
\{\text{rdf : type, fname, lname}\} & \quad \text{subsumes}
\end{align*}
\]
Basic Concepts

Signature Subsumption Graph \( \mathcal{G} = (\sum_G, \subseteq) - (def \ 4) \)

Properties

- directed acyclic graph
- vertices from graph signature set
- edges based on subsumption relationship

\( \sum_G \subseteq \{ rdf : type \} \)

\( \{ rdf : type, fname, lname \} \) subsumes \( \{ rdf : type, label, fname, lname \} \)
RDF Graph into Subsumption Graph

Person rdf:type rdf:Class

p₁ rdf:type

fname lname

label

p₃ rdf:type

fname lname

\{ \text{rdf : type} \}

\text{subsumes}

\{ \text{rdf : type, fname, lname} \}

\text{subsumes}

\{ \text{rdf : type, label, fname, lname} \}
Subsumtion Graph to Tables

\{ \text{rdf : type} \} \quad \text{subsumes} \quad \{ \text{rdf : type, fname, lname} \} \quad \text{subsumes} \quad \{ \text{rdf : type, label, fname, lname} \}
Basic Concepts

Signature Table – (def 5)

– for a signature \((p_1, \ldots, p_n)\) in the subsumption graph construct a two dimensional table

• with columns (rdf:about, \(p_1, \ldots, p_n\))
Signature Table – (def 5)

- for a signature \((p_1, \ldots, p_n)\) in the subsumtion graph construct a two dimensional table

\[
\begin{array}{|l|l|}
\hline
\text{rdf:about} & \text{rdf:type} \\
\hline
\end{array}
\]

\[
\begin{array}{|l|l|l|}
\hline
\text{rdf:about} & \text{rdf:type} & \text{fname} & \text{lname} \\
\hline
\end{array}
\]
Basic Concepts

Signature Table – (def 5)

– each subject (corresponding to the signature) becomes a row in the respective signature table

\[
\{ \text{rdf : type} \}
\]

<table>
<thead>
<tr>
<th>rdf:about</th>
<th>rdf:type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>rdfs:Class</td>
</tr>
</tbody>
</table>

\[
\{ \text{rdf : type, fname , lname} \}
\]

<table>
<thead>
<tr>
<th>rdf:about</th>
<th>rdf:type</th>
<th>fname</th>
<th>lname</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>Person</td>
<td>“Peter”</td>
<td>“Miller”</td>
</tr>
</tbody>
</table>
## Basic Concepts

**Signature Table Set** – *(def 6)*

\[
\mathcal{T}_G = \{ \tau_G(s) \mid s \in \sum_G \}
\]

<table>
<thead>
<tr>
<th>rdf:about</th>
<th>rdf:type</th>
<th>fname</th>
<th>Iname</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_3</td>
<td>Person</td>
<td>“Peter”</td>
<td>“Miller”</td>
</tr>
<tr>
<td>p_1</td>
<td>Person</td>
<td>“Michael”</td>
<td>“Sintek”</td>
</tr>
</tbody>
</table>
RDF Graph into Signature Tables

Person

p₁

p₃

fname

lname

label

rdf:type

fname

lname

rdf:type

fname

 lname

...
RDF Graph Operations

Projection (def 7)

\[ \pi_{p=(p_1, \ldots, p_n)}(G) = \bigcup \pi_p(t) \]

for all \( t = \tau_G(s) \) with \( s \in \Sigma_G \land p \subseteq s \)
RDF Graph Operations

Projection (def 7)

\[ \dot{\pi}(p_1, \ldots, p_n)(G) = \bigcup_{t} \pi(p_1, \ldots, p_n)(t) \]

- graph projection → database projections

- sufficient to only have graph projection
  - full range of relational operators are allowed on signature tables
  - all relational algebra expressions can be reformulated to have projections occurring first
RDF Graph Operations

Projection–Selection (*def 8*)

- add selection operation for projection efficiency
RDF Graph Operations

Projection–Selection (\textit{def 8})

\[
\left[ \pi \sigma \right]_p^C (G) = \pi_p \cup (\sigma_C \circ \pi_{p'}) (t)
\]

for all \( t = \tau_G(s) \) with \( s \in \sum G \)
and \( p' = p \cup \text{properties}(C) \)
and \( C = \text{select conditions} \)
RDF Graph Operations

Projection–Selection (def 8)

\[ [\pi \cdot \sigma]_p^C (G) = \pi_p \cup (\sigma_C \circ \pi_{p'}) (t) \]

- add selection operation for projection efficiency

\[ p \subseteq s \quad \text{and} \quad p' \subseteq s \]

\[ p = \{ p_1, \ldots, p_n \} \quad \text{and} \quad p' = p \cup \text{properties}(C) \]

only consider signature tables with signatures that are subsumed by the properties occurring in the operators
Sample Queries

Example 1

*return first name and last name for all persons*

\[ R = \left[ \pi \sigma \right]^{\text{rdf:type = Person}}_{\text{firstName, lastName}}(P) \]
Sample Queries

Example 1

return first name and last name for all persons

\[ R = \pi_{\text{firstName}, \text{lastName}} \sigma_{\text{rdf:type} = \text{Person}} (P) \]

Example 2

find first name, email address, and homepage for the person with last name “Smith”

\[ R = \pi_{\text{firstName}, \text{email}, \text{homepage}} \sigma_{\text{rdf:type} = \text{Person} \land \text{lastName} = 'Smith'} (P) \]
Evaluation

Dataset – swirl-SiteMoviesIMDB.rdf

- 3,587,064 triples, 286,273 subjects → 4,708 signature tables

Compared against

- Apache Jenna
- Sesame
Evaluation

Dataset – swirl-SiteMoviesIMDB.rdf

- 3,587,064 triples, 286,273 subjects → 4,708 signature tables

<table>
<thead>
<tr>
<th>tuples per table</th>
<th>tuples</th>
<th>tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–10</td>
<td>8941</td>
<td>4137</td>
</tr>
<tr>
<td>11–100</td>
<td>15375</td>
<td>506</td>
</tr>
<tr>
<td>101–1000</td>
<td>13730</td>
<td>62</td>
</tr>
<tr>
<td>1001–10000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10001–100000</td>
<td>117288</td>
<td>2</td>
</tr>
<tr>
<td>100001–1000000</td>
<td>130939</td>
<td>1</td>
</tr>
</tbody>
</table>
## Evaluation

**Dataset – swirl-SiteMoviesIMDB.rdf**

- 3,587,064 triples, 286,273 subjects → 4,708 signature tables

<table>
<thead>
<tr>
<th>tuples per table</th>
<th>tuples</th>
<th>tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–10</td>
<td>8941</td>
<td>4137</td>
</tr>
<tr>
<td>11–100</td>
<td>15375</td>
<td>506</td>
</tr>
<tr>
<td>1001–10000</td>
<td>0</td>
<td>62</td>
</tr>
<tr>
<td>10001–100000</td>
<td>117288</td>
<td></td>
</tr>
<tr>
<td>100001–1000000</td>
<td>130939</td>
<td></td>
</tr>
</tbody>
</table>

*most of the data resides in two tables*
Evaluation

Query Performance – swirl-SiteMoviesIMDB.rdf

Query 1 [select] return some interesting properties of all movies

Query 2 [join] find names for persons casted in movies

Query 3 [path expression along one property]
    find persons playing in movies three cast hops separated from Kevin Bacon

Query 4 [same as] find movies with same title and return some useful properties on them
Evaluation

Query Performance – swirl-SiteMoviesIMDB.rdf

Query 1: **select** return some interesting properties of all movies

Query 2: **join** find names for persons casted in movies

Query 3: **path expression along one property**

find persons playing in movies three cast hops separated from Kevin Bacon

Query 4: **same as** find movies with same title and return some useful properties on them
Evaluation

Query Performance – *swirl-SiteMoviesIMDB.rdf*

<table>
<thead>
<tr>
<th></th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>memory</td>
<td>time</td>
<td>memory</td>
</tr>
<tr>
<td>RDFBroker</td>
<td>70ms</td>
<td>4MB</td>
<td>1200ms</td>
<td>63MB</td>
</tr>
<tr>
<td>Jena</td>
<td>4300ms</td>
<td>82MB</td>
<td>8700ms</td>
<td>26MB</td>
</tr>
<tr>
<td>Sesame</td>
<td>1400ms</td>
<td>24MB</td>
<td>2200ms</td>
<td>46MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>260ms</td>
<td>4MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>70ms</td>
<td>3MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50ms</td>
<td>2MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>160ms</td>
<td>10MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Query 1 *[select]* return some interesting properties of all movies

Query 2 *[join]* find names for persons casted in movies

Query 3 *[path expression along one property]*
find persons playing in movies three cast hops separated from Kevin Bacon

Query 4 *[same as]* find movies with same title and return some useful properties on them
Evaluation

Query Performance – swirl-SiteMoviesIMDB.rdf

|        | Query 1 |  | Query 2 |  | Query 3 |  | Query 4 |  |
|--------|---------|  |---------|  |---------|  |---------|  |
|        | time    | memory | time    | memory | time    | memory | time    | memory |
| RDFBroker | 70ms    | 4MB     | 1200ms  | 63MB    | 260ms   | 4MB     | 160ms   | 10MB    |
| Jena   | 4300ms  | 82MB    | 8700ms  | 26MB    | 70ms    | 3MB     | -       | -       |
| Sesame | 1400ms  | 24MB    | 2200ms  | 46MB    | 50ms    | 2MB     | -       | -       |

Query 3 [path expression along one property]
find persons playing in movies three cast hops separated from Kevin Bacon

RDFBroker has to traverse thousands of tables multiple times
Evaluation

Query Performance – *swirl-SiteMoviesIMDB.rdf*

<table>
<thead>
<tr>
<th></th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>memory</td>
<td>time</td>
<td>memory</td>
</tr>
<tr>
<td>RDFBroker</td>
<td>70ms</td>
<td>4MB</td>
<td>1200ms</td>
<td>63MB</td>
</tr>
<tr>
<td>Jena</td>
<td>4300ms</td>
<td>82MB</td>
<td>8700ms</td>
<td>26MB</td>
</tr>
<tr>
<td>Sesame</td>
<td>1400ms</td>
<td>24MB</td>
<td>2200ms</td>
<td>46MB</td>
</tr>
<tr>
<td></td>
<td>260ms</td>
<td>4MB</td>
<td>160ms</td>
<td>10MB</td>
</tr>
<tr>
<td></td>
<td>70ms</td>
<td>3MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>50ms</td>
<td>2MB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Query 3** [path expression along one property]

*find persons playing in movies three cast hops separated from Kevin Bacon*

**Query 4** [same as] *find movies with same title and return some useful properties on them*

*queries requiring a lot of joins are taxing traditional database schema*
Conclusion

RDFBroker

- an RDF store
  - backed up by a database
  - utilizes properties *signature* to define storage tables
- schema independent
- comparable performance to
  - hand-coded schema based storage
Discussion
Discussion

RDF not Suitable
Signature Table Selection
Signature Tables?
Jenna & Sesame Table Models
No Inference
RDF not Suitable

Not suitable for applications & databases

- triple notation is the lowest common denominator
- triple notation does not lack expressivity

scheduled(cs422, 2, 1030, cc208)

b123 rdf:type booking
b123 course cs422
b123 section 2
b123 starts 1030
b123, room, cc208

RDF not Suitable

Not suitable for applications & databases

- application data structures are object oriented and not statement oriented

- RDF is not statement oriented rather RDF graphs are very similar to object graphs

- relational databases do not offer a natural mapping of application object graphs either
  - *OO inheritance*
Graph Vs. Relational Database

Observation

- graphs (unlike relational databases) don't have fixed schema
- an RDF graph mutates with each added triple
Signature Table Selection

Every graph query requires to lookup the corresponding signature tables

- RDFBroker produces thousands of signature tables
Signature Table Selection

Every graph query requires to lookup the corresponding signature tables
Signature Table Selection

Every graph query requires to lookup the corresponding signature tables

- add an artificial root
- starting at the root, find all minimal signatures subsumed by $p$
- collect all signatures using depth first walk
Signature Table Selection

Every graph query requires to lookup the corresponding signature tables

- add an artificial root
- starting at the root, find all minimal signatures subsumed by $p$
- collect all signatures using depth first walk
Signature Table Selection

Every graph query requires to lookup the corresponding signature tables

- add an artificial root
- starting at the root, find all minimal signatures subsumed by $p$
- collect all signatures using depth first walk
Signature Tables?

Column Index Performance

“there is an index for each column in a signature table, currently realized as a hash table”

- column indices have an associated cost

“there is a trade-off between the speed of retrieving data from a table and the speed of updating the table”*

*http://docs.oracle.com/cd/B10501_01/server.920/a96521/indexes.htm#310
Signature Tables?

RDF inserts/update → Schema Mutation

– adding new properties → new subsumtion graph structure

– change in signature →
  • potentially expensive database operations (such as table migrations)
  • has a direct impact on the initial signature table lookup
Signature Tables?

Subject Lookup

- a subject \( s \) is mapped to a signature table based on the property signature \( p \) of \( s \)

but knowing \( p \) implies that all properties of \( s \) are known

what if only a subsect of \( p \) is known?
Signature Tables?

Inference & Subject Lookup

\textit{inference makes implied information explicit}

- an inference process will result in additional properties
  - certain RDF properties can be implied
  - the implied properties were not part of the initial signature
Signature Tables?

Inference & Subject Lookup

\[ \Sigma_G (\text{Mary}) = \{ \text{hasChild} \} \]

Diagram:
- Node: Mary
- Edge: hasChild
- Edge: hasChild
Signature Tables?

Inference & Subject Lookup

\[ \Sigma_G (\text{Mary}) = \{ \text{hasChild} \} \]
Signature Tables?

Inference & Subject Lookup

\[ \sum_G(Mary) = \{ \text{hasChild}, \text{isParentOf} \} \]
Signature Tables?

Inference & Subject Lookup

\[ \Sigma_G(Mary) = \{ \text{hasChild}, \text{isParentOf}, \text{isGrandParentOf} \} \]

![Diagram showing relationships between Mary and other entities with relationships like hasChild, isParentOf, and isGrandParentOf.](image-url)
Signature Tables?

Inference & Subject Lookup

\[ \Sigma_G(Mary) = \{ hasChild, isParentOf, isGrandParentOf \} \]

\[ \Sigma_G(isParentOf) = \{ ... , rdf:subProperty \} \]
**Signature Tables?**

**Table Imbalance**

- 87% of the data was contained in two tables.

<table>
<thead>
<tr>
<th>tuples per table</th>
<th>tuples</th>
<th>tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–10</td>
<td>8941</td>
<td>4137</td>
</tr>
<tr>
<td>11–100</td>
<td>15375</td>
<td>506</td>
</tr>
<tr>
<td>101–1000</td>
<td>13730</td>
<td>62</td>
</tr>
<tr>
<td>1001–10000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10001–100000</td>
<td>117288</td>
<td>2</td>
</tr>
<tr>
<td>1000001–10000000</td>
<td>130939</td>
<td>1</td>
</tr>
</tbody>
</table>
Merging Signatures

Algorithm

- merge small adjacent signatures tables based on subsumtion relationship using some threshold value.

“resulting signature subsumption graphs are often very similar to user defined schemas......”
Jenna & Sesame Table Models*

Jenna

Sesame

*http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/
Projection Operation Only?

Only Projection Operation is Required?

- “all relational algebra expressions can be reformulated to have projections occurring first” ??
No Select (σ) Graph Operator?

- “we do not define an operator σ on RDF graphs since signature tables that would naturally be involved in a single selection are of varying arity”
Extra Slides
Objective – Main Idea

Map RDF triples into database tables

Design Goals

- schema independence
- 'efficient' import/export of RDF data
- 'efficient' access from applications
### Database Projection

**Person**

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry</td>
<td>34</td>
<td>80</td>
</tr>
<tr>
<td>Sally</td>
<td>28</td>
<td>64</td>
</tr>
<tr>
<td>George</td>
<td>29</td>
<td>70</td>
</tr>
<tr>
<td>Helena</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Peter</td>
<td>34</td>
<td>80</td>
</tr>
</tbody>
</table>

\[ \pi_{AGE,WEIGHT}(Person) \]

<table>
<thead>
<tr>
<th>AGE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>80</td>
</tr>
<tr>
<td>28</td>
<td>64</td>
</tr>
<tr>
<td>29</td>
<td>70</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
</tr>
</tbody>
</table>

*http://en.wikipedia.org/wiki/Projection_%28relational_algebra%29*
# Database Selection*


<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry</td>
<td>34</td>
<td>80</td>
</tr>
<tr>
<td>Sally</td>
<td>28</td>
<td>64</td>
</tr>
<tr>
<td>George</td>
<td>29</td>
<td>70</td>
</tr>
<tr>
<td>Helena</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Peter</td>
<td>34</td>
<td>80</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{AGE} \geq 34}(\text{Person}) \]

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry</td>
<td>34</td>
<td>80</td>
</tr>
<tr>
<td>Helena</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>Peter</td>
<td>34</td>
<td>80</td>
</tr>
</tbody>
</table>

\[ \pi_{\text{AGE} = \text{WEIGHT}}(\text{Person}) \]

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helena</td>
<td>54</td>
<td>54</td>
</tr>
</tbody>
</table>