DATABASES

INTRODUCTION TO DATA SCIENCE
TIM KRASKA
OTHER ANNOUNCEMENTS

Want to get involved in research?

We are offering several independent studies and summer research internship.

Sign-up available on: http://database.cs.brown.edu/
or directly: http://tinyurl.com/zzznf92

Possible Topics:

- Infiniband
- Tupleware
- Interactive Data Exploration
Do you want to drink from the fire hose??? Then take my CS227 class.
So far:

Database System Concepts
Sixth Edition by Silberschatz.

Pieces of chapters 1, 2, 3, 6, 7 and (18)
DATABASES FOR DATA SCIENTIST

1. Requirement Engineering
2. Conceptual Modeling
3. Logical & Physical Modeling
4. Ask Questions

- Book of duty
- Conceptual Design (ER)
- Logical design (schema)
- Physical design (index, hints)
CLICKER QUESTION I

- A customer can have several orders
- An order belongs to a single customer
- Every order has exactly one shipping method (e.g., Post, FedEx, UPS,...)
A customer can have several orders
An order belongs to a single customer
Every order has exactly one shipping method (e.g., Post, Fexed, UPS,...)
PROBLEM

- **You are the new Data Scientist at Evil Market**
- Evil Market is tracking all customer purchases with their membership or credit card
- They also have data about their customers (estimated income, family status,…) from surveys they have done in the past
- Recently, they are trying to improve their image for young mothers
- As a start they want to know the following information for mothers under 30 for 2015:
  - How much do they spend at Evil Market?
  - How does this compare to all customers under 30?
  - What are their favorite products?
  - Did they spend more in 2015 than in 2014?

Your first project: Design the schema for Evil Market to analyze Evil Market’s purchase logs!
STAR SCHEMA

Fact Table
- Shop ID
- Customer ID
- Profit
- Volume
- Etc…

City
- City ID
- Name
- Population

Shop:
- Shop ID

Product
- Product ID
- Type ID
- Name
- Brand ID

Time
- Date ID
- Month ID

Customer
- Customer ID
- Customer Group ID
- Name
Databases for Data Scientist

Requirement Engineering

Conceptual Modeling

Logical & Physical Modeling

Ask Questions

- Logical design (schema)
- Physical design (index, hints)

Book of duty

Conceptual Design (ER)
SQL: RELATIONAL ALGEBRA
FORMAL DEFINITION OF REL. ALGEBRA

Atoms (basic expressions)
• A relation in the database
• A constant relation

Operators (composite expressions)
• Selection: $\sigma (E_1)$
• Projection: $\Pi (E_1)$
• Cartesian Product: $E_1 \times E_2$
• Rename: $\rho_V(E_1), \rho_{A \leftarrow B}(E_1)$
• Union: $E_1 \cup E_2$
• Minus: $E_1 - E_2$
CLOSURE PROPERTY / COMPOSABILITY
Professor(Person-ID:integer, Name:varchar(30), Level:varchar(2))
Student(Student-ID:integer, Name:varchar(30), Semester:integer)
Lecture(Course-ID:varchar(10), Title:varchar(50), CP:float)
Gives(Person-ID:integer, Course-ID:varchar(10))
Attends(Student-ID:integer, Course-ID:varchar(10))
Tests(Student-ID:integer, Course-ID:varchar(10), Person-ID:integer, Grade:char(2))
SELECTION AND PROJECTION

Professor(Person-ID:integer, Name:varchar(30), Level:varchar(2))
Student(Student-ID:integer, Name:varchar(30), Semester:integer)

Selection

\[ \sigma_{\text{Semester} > 10} (\text{Student}) \]

<table>
<thead>
<tr>
<th>Student-ID</th>
<th>Name</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>24002</td>
<td>Xenokrates</td>
<td>18</td>
</tr>
<tr>
<td>25403</td>
<td>Jonas</td>
<td>12</td>
</tr>
</tbody>
</table>

Projection

\[ \Pi_{\text{Level}} (\text{Professor}) \]

<table>
<thead>
<tr>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
</tr>
<tr>
<td>AP</td>
</tr>
</tbody>
</table>
CARTESIAN PRODUCT

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
<td></td>
</tr>
</tbody>
</table>

\[ L \times R = \begin{array}{cc}
A & B & C & D & E \\
\hline
a₁ & b₁ & c₁ & d₁ & e₁ \\
a₁ & b₁ & c₁ & d₂ & e₂ \\
a₂ & b₂ & c₂ & d₁ & e₁ \\
a₂ & b₂ & c₂ & d₂ & e₂ \\
\end{array} \]
CARTESIAN PRODUCT (CTD.)

Professor $\times$ Attends

<table>
<thead>
<tr>
<th>Professor</th>
<th>Attends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person-ID</td>
<td>Name</td>
</tr>
<tr>
<td>2125</td>
<td>Ugur</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2125</td>
<td>Ugur</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2137</td>
<td>Jeff</td>
</tr>
</tbody>
</table>

- Huge result set ($n \times m$)
- Usually only useful in combination with a selection (-> Join)
**NATURAL JOIN**

Two relations:

- \( R(A_1, \ldots, A_m, B_1, \ldots, B_k) \)
- \( S(B_1, \ldots, B_k, C_1, \ldots, C_n) \)

\[
R \bowtie S = \prod_{A_1, \ldots, A_m, B_1, \ldots, B_k, C_1, \ldots, C_n} \left( \sigma_{R.B_1 = S.B_1} \land \ldots \land R.B_k = S.B_k \right) (R \times S)
\]

<table>
<thead>
<tr>
<th>( R \bowtie S )</th>
<th>( R - S )</th>
<th>( R \cap S )</th>
<th>( S - R )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( A_1 )</td>
<td>( A_2 )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>
### THREE-WAY NATURAL JOIN

(\text{Student} \bowtie \text{attends}) \bowtie \text{Lecture}

<table>
<thead>
<tr>
<th>Student-ID</th>
<th>Name</th>
<th>Semester</th>
<th>Course-NR</th>
<th>Title</th>
<th>CP</th>
<th>Person-ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>26120</td>
<td>Fichte</td>
<td>10</td>
<td>CS1951</td>
<td>Intro to Data Science</td>
<td>2</td>
<td>9999</td>
</tr>
<tr>
<td>27550</td>
<td>Jonas</td>
<td>12</td>
<td>CS18</td>
<td>Programming</td>
<td>2</td>
<td>2134</td>
</tr>
<tr>
<td>28106</td>
<td>Carnap</td>
<td>3</td>
<td>CS19</td>
<td>More Programming</td>
<td>3</td>
<td>2126</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

...
THETA-JOIN

Two Relations:

- $R(A_1, ..., A_n)$
- $S(B_1, ..., B_m)$

\[ R \bowtie_{\theta} S = \sigma_{\theta} (R \times S) \]
# JOIN VARIANTS

- **natural join**

<table>
<thead>
<tr>
<th>L</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
</tr>
<tr>
<td>2nd</td>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
</tr>
<tr>
<td>2nd</td>
<td>c₃</td>
<td>d₂</td>
<td>e₂</td>
</tr>
</tbody>
</table>

Result

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>d₁</td>
</tr>
</tbody>
</table>

- **left outer join**

<table>
<thead>
<tr>
<th>L</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
</tr>
<tr>
<td>2nd</td>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
</tr>
<tr>
<td>2nd</td>
<td>c₃</td>
<td>d₂</td>
<td>e₂</td>
</tr>
</tbody>
</table>

Result

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>d₁</td>
</tr>
<tr>
<td>2nd</td>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
<td>-</td>
</tr>
</tbody>
</table>
JOIN VARIANTS

- right outer join

<table>
<thead>
<tr>
<th>L</th>
<th>R</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>c₃</td>
<td>d₂</td>
<td>e₂</td>
</tr>
</tbody>
</table>
JOIN VARIANTS

- (full) outer join

\[
\begin{array}{ccc}
\text{L} & \text{R} & \text{Result} \\
A & B & C & C & D & E \\
a_1 & b_1 & c_1 & c_1 & d_1 & e_1 \\
a_2 & b_2 & c_2 & c_3 & d_2 & e_2 \\
\end{array}
\]

- left semi join

\[
\begin{array}{ccc}
\text{L} & \text{R} & \text{Result} \\
A & B & C & C & D & E \\
a_1 & b_1 & c_1 & c_1 & d_1 & e_1 \\
a_2 & b_2 & c_2 & c_3 & d_2 & e_2 \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{L} & \text{R} & \text{Result} \\
A & B & C & C & D & E \\
a_1 & b_1 & c_1 & c_1 & d_1 & e_1 \\
a_2 & b_2 & c_2 & - & - & - \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{L} & \text{R} & \text{Result} \\
A & B & C & C & D & E \\
a_1 & b_1 & c_1 & c_3 & d_2 & e_2 \\
a_2 & b_2 & c_2 & - & - & - \\
\end{array}
\]
JOIN VARIANTS

- right semi join

\[
\begin{array}{ccc}
A & B & C \\
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
\end{array} \quad \times \quad \begin{array}{ccc}
C & D & E \\
c_1 & d_1 & e_1 \\
c_3 & d_2 & e_2 \\
\end{array} = \begin{array}{ccc}
\text{Resultat} \\
C & D & E \\
c_1 & d_1 & e_1 \\
\end{array}
\]
Renaming of relation names

- Needed to process self-joins and recursive relationships
- E.g., two-level dependencies of lectures ("grandparents")

\[
\Pi_{\text{course-id}, \text{prerequisite}} (\sigma_{\text{course-id}=\text{CS1951A}} (\rho_{\text{Requires}} (L1, L2)))
\]

<table>
<thead>
<tr>
<th>course-id</th>
<th>prerequisite</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1951A</td>
<td>CS160</td>
</tr>
<tr>
<td>CS1951A</td>
<td>CS320</td>
</tr>
<tr>
<td>CS2270</td>
<td>CS1270</td>
</tr>
<tr>
<td>CS1270</td>
<td>CS160</td>
</tr>
</tbody>
</table>
SET DIFFERENCE (−)

Notation: \( Relation_1 - Relation_2 \)

R - S valid only if:

1. \( R, S \) have same number of columns (arity)
2. \( R, S \) corresponding columns have same domain (compatibility)

Example:

\[
(\Pi_{\text{bname}} (\sigma_{\text{amount} \geq 1000} (\text{loan}))) - (\Pi_{\text{bname}} (\sigma_{\text{balance} < 800} (\text{account})))
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{bname} & \text{lno} & \text{amount} \\
\hline
\text{Downtown} & L-17 & 1000 \\
\text{Redwood} & L-23 & 2000 \\
\text{Perry} & L-15 & 1500 \\
\text{Downtown} & L-14 & 500 \\
\text{Perry} & L-16 & 300 \\
\hline
\end{array}
\quad
\begin{array}{|c|c|c|}
\hline
\text{bname} & \text{acct_no} & \text{balance} \\
\hline
\text{Mianus} & A-215 & 700 \\
\text{Brighton} & A-201 & 900 \\
\text{Redwood} & A-222 & 700 \\
\text{Brighton} & A-217 & 850 \\
\hline
\end{array}
\]

Result:
\[
\begin{array}{|c|}
\hline
\text{bname} \\
\hline
\text{Mianus} \\
\text{Redwood} \\
\hline
\end{array}
\quad
\begin{array}{|c|}
\hline
\text{bname} \\
\hline
\text{Downtown} \\
\text{Redwood} \\
\hline
\end{array}
\quad
\begin{array}{|c|}
\hline
\text{bname} \\
\hline
\text{Downtown} \\
\text{Perry} \\
\hline
\end{array}
\]

= (A) (B) (C)
INTERSECTION

\( \Pi_{\text{Person-ID}}(\text{Lecture}) \cap \Pi_{\text{Person-ID}}(\sigma_{\text{Level}=\text{FP}}(\text{Professor})) \)

Only works if both relations have the same schema

• Same attribute names and attribute domains

Intersection can be simulated with minus:

\( R \cap S = R - (R - S) \)

Union works similarly...
Codd’s Theorem

3 Languages:
• Relational Algebra
• Tuple Relational Calculus (safe expressions only)
• Domain Relational Calculus (safe expressions only)
are equivalent.

Impact of Codd’s theorem:
• SQL is based on the relational calculus
• SQL implementation is based on relational algebra
• Codd’s theorem shows that SQL implementation is correct and complete.
NOT COVERED

Set Division
Aggregate Functions
Codd´s Proof
...
