Tutorial 7 — Data Mining and Transactions

Richard Wong
rk2wong@edu.uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

March 12, 2018

Exercise 7-1

What is a data warehouse, and why is it useful to have one?

Exercise 7-1 Solution

What is a data warehouse, and why is it useful to have one?

- Maintain OLTP responsiveness by having separate hardware handle OLAP queries
- A data warehouse can be optimized for read access
- Reshape (denormalize) OLTP schema into star/snowflake schema to simplify OLAP queries
- Star/snowflake schemas enable cubing operations on data (look up "OLAP cube")
- Most OLAP queries deal in aggregates, which could benefit from column-oriented storage
- and more...

Suppose we are trying to predict the value *Wait*.
Between attributes *Pat* and *Type*, which is better to split a decision node on?

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Exercise 7-2 Solution (1/3)

Let's use entropy (I) as the metric that we want to minimize.

We have two choices of attributes (*Pat* and *Type*), and we want to pick the one that maximizes information gain. So first we should find out the entropy of the full data set S.

$$
\begin{aligned}
I(S) &= -\sum_{i=1}^{\{T,F\}} p_i \log_2(p_i) \\
&= -p_T \log_2(p_T) - p_F \log_2(p_F) \\
&= -(\frac{6}{12}) \log_2(\frac{6}{12}) - (\frac{6}{12}) \log_2(\frac{6}{12}) \\
&= -(\frac{1}{2})(-1) - (\frac{1}{2})(-1) \\
&= \frac{1}{2} + \frac{1}{2} \\
&= 1
\end{aligned}
$$

Exercise 7-2 Solution (2/3)

So $I(S) = 1$.

$$
\begin{aligned}
I(S_{Pat}) &= P_{Pat=None} I(S_{Pat=None}) + P_{Pat=Some} I(S_{Pat=Some}) + P_{Pat=Full} I(S_{Pat=Full}) \\
&= \frac{2}{12}(0) + \frac{4}{12}(0) + \frac{6}{12} I(S_{Pat=Full}) \\
&= \frac{1}{2}(0.918) \\
IG_{Pat} &= I(S) - I(S_{Pat}) \\
&= 1 - \frac{1}{2}(0.918) \\
&= 0.541
\end{aligned}
$$

Exercise 7-2 Solution (3/3)

$$\begin{aligned}
I(S_{Type}) &= P_{Type=French} I(S_{Type=French}) + P_{Type=Italian} I(S_{Type=Italian}) \\
&+ P_{Type=Thai} I(S_{Type=Thai}) + P_{Type=Burger} I(S_{Type=Burger}) \\
&= \frac{1}{4}(1) + \frac{1}{4}(1) + \frac{1}{4}(1) + \frac{1}{4}(1) \\
&= 1 \\
IG(S_{Type}) &= I(S) - I(S_{Type}) \\
&= 1 - 1 \\
&= 0
\end{aligned}$$

So $IG(S_{Pat}) = 0.541$, $IG(S_{Type}) = 0$.
$IG(S_{Pat}) > IG(S_{Type})$, so *Pat* is the better attribute to split on.

Exercise 7-3

What are the ACID transaction properties, and what can a database do to ensure each one?

Exercise 7-3 Solution (1/2)

The ACID properties:

- Atomicity: the effects of a transaction either fully take place, or none at all.
- Consistency preservation: the state of a database before and after a transaction is consistent.
- Isolation: transactions can operate unaware of concurrent transactions.
- Durability: transaction completion guarantees persistence (i.e. write to disk or stable storage, backup, etc.)

Exercise 7-3 Solution (2/2)

How do we ensure each one?

- Atomicity: keep a sequential log of each task that executes in a transaction. Each log entry should contain sufficient data to roll back. In addition, only allow execution of recoverable transaction schedules.
- Consistency: abort transactions that would violate constraints.
- Isolation: this can be done to varying degrees. It is up to the transaction manager to uphold isolation guarantees, which the DBA may decide on.
- Durability: never report that a transaction is committed until it is persisted.

Exercise 7-4

Distinguish between the following:

1. a serial schedule
2. a serializable schedule
3. a conflict-serializable schedule

Exercise 7-4 Solution

1. serial: aside from the first transaction in the schedule, each transaction only starts after the previous has committed.
2. serializble: the schedule is *equivalent* to a serial schedule using the same transactions, for some definition of *equivalent*.
3. conflict-serializable: the schedule is *conflict-equivalent* to a serial schedule using the same transactions.

If you can swap two consecutive *non-conflicting* operations (belonging to different transactions) in a schedule, then the schedules before and after the swap are considered to be *conflict-equivalent*.
Two operations are *conflicting* iff at least one of them is a write, and they both operate on the same data.

Exercise 7-5

Is the following schedule conflict-serializable?
If not, how can we make it conflict-serializable?

T1	r(x)		w(y)				
T2		r(y)			r(x)		
T3				w(x)		r(x)	

Exercise 7-5 Solution

T1	r(x)		w(y)			
T2		r(y)			r(x)	
T3				w(x)		r(x)

This schedule is not conflict-serializable. To show that there is no sequence of transformations that can take the given schedule to a serial schedule, we use the concept of a precedence graph.

In this graph, we create a *node for each transaction*, then proceed to add edges.

A directed edge from transaction X to transaction Y in the precedence graph means that *X precedes Y*. That means that whatever serial schedule we come up with needs to execute everything in X befor everything in Y.

We add an edge from X to Y where an operation in X occurs before and conflicts with an operation in Y.

A cycle in the precedence graph means that no serial ordering of the transactions will be conflict-equivalent with the given schedule. Try creating the graph as an exercise.

To make the schedule conflict-serializable, remove nodes (i.e. abort transactions) until there are no cycles.