# 1 Valid annotation constraints

Among the *key*, *distinct*, and *identifying* constraints, there are correlations between "key yes" and "distinct yes".

The following example gives us a hint of invalid annotations. Given the data in Table 1. The following annotation is *invalid*.

**observation** "o1" distinct yes
    **entity** "Plot"
    **measurement** "m1" key yes
        **characteristic** "EntityName"
        **standard** "Nominal"
    **measurement** "m2"
        **characteristic** "area"
        **standard** "sqft"
    $\cdots$ Not finished....

According to the annotation, if two plots have the same value on EntityName, they represent the same plot observation. Obviously, it has probelem to interprete the data in Table 1 with this annotatioin. E.g., the first and second row catching information about plot with EntityName $A$. Accoring to the annotation, they should be the same plot obervation. However, the data shows that this one plot has two different areas 1.0 and 1.1, so, there is confusion here.

| plt | area | spp | dbh |
|-----|------|------|------|
| A | 1.0 | piru | 35.8 |
| A | 1.1 | piru | 36.2 |
| B | 2.0 | piru | 33.2 |
| B | 2.0 | abba | 34 |

Table 1: Dataset

**From HP:** There are several ways to let the data comply with the annotaion without confusion. Maybe too detailed, I omit this here.

Here are some very basic rules to restrict the correlations among these constraints.

- "Distinct yes" on an observation type implies that all its measurements collectively form a key of this observation type. In the above example, we can specify "distinct yes" on "$o_1$" without *explicitly* denote "key yes" on the measurements $m_1$ and $m_2$. The semantic meaning is that $(m_1, m_2)$ collectively form key measurement for $o_1$.

- Logically, we can specify "Distinct no" on an observation type and specify "key yes" on all of its measurements. In this case, when two rows (for the observation type) have the same values for these key measurements, they represent the different observation instances. Such a case means that there may exist other "variables" that can distinguish one observation instance from the other, but this dataset does not catch this variable (**HP: is "variable" the right word to use?** ). So, in real application, we do not encourage such situation to appear.

In our algorithm, we assume that the annotations on the dataset are valid, i.e., they comply with the rules among these constraints.

**From HP:** How detailed to discuss this?

# 2   Data materialization algorithm

The **MaterializeDB** algorithm catches the *key*, *distinct* and *identifying* constraints in the annotation during the materialization process. The input of the algorithm is the *Dataset* and the annotations $A.*$ on it. Each row in the input dataset represents the information related to one or more observations and their contexts. The input $A.*$ represents the annotation information. More specifically,

- A.MeasType $= \{\langle \underline{MeasTypeId}, ObsTypeId, CharType, StdType, ProtType, Precision, isKey \rangle\}$;

- A.ObservationType $= \{\langle \underline{ObsTypeId}, EntTypeId, isDistinct \rangle\}$
  Note that, we hide the "AnnotId" in this schema, which denotes the resource the annotation is on. We do not include this because the algorithm focuses on dealing with annotations on one resource. This way, we can simplify the description.

- A.ContextType $= \{\langle \underline{ObsTypeId, ContextObsTypeId, RelType}, isIdentify \rangle\}$

- A.Map $= \{\langle \underline{MeasTypeId, ResAttribute, Cond}, Val \rangle\}$

The output of the algorithm is a set of materialized tables represented in the OBOE model and denoted by $OBOE.*$. In detail,

- $OBOE.Observation = \{\langle \underline{ObsId}, EntId \rangle\}$ keeps all the observation instances;

- $OBOE.Measurement = \{\langle \underline{MeasId}, ObsId, MeasType, Val \rangle\}$ for all the measurement instances;

- $OBOE.Entity = \{\langle \underline{EntId}, EntType \rangle\}$ for all the entity instances;

- $OBOE.Context = \{\langle \underline{ObsId, ConextObsId, ContextType} \rangle\}$ for all the context instances;

Algorithm 1 shows the framework of our algorithm. In this algorithm, we maintain two intermediate index structures ($EntIdx$ and $ObsIdx$) to keep track of the distinct entity and observation instances. $ObsIdx$ is the index structure maintained for the observation instances whose types are specified with *distinct yes*. This index maintains the mapping from observation type and key values to its corresponding observation instance id. Obviously, only when an observation type is specified with *distinct yes*, (i.e., we want to keep track of the same observation instances) we need to maintain their instances in this index.

The key values can be calculated in three different cases. In the first case where only one measurement of an observation type is specified with "key yes", the key value is the value of this "key" measurement. In the second case, several measurement types are marked with "key yes", the key value is the combined instance value of these several measurement types. In the third case, some context of this object type is marked with "identifying yes", the key value is the combination of the instance values of its own key measurement types and the instance values of its context observation's key measurements.

$Entidx$ is the index structure for tracking the distinct entities. In case that some measurement type(s) of an observation type is/are specified with "key yes", if different observation instances have the same value on these key measurements, semantically, we interpret that they are of the same entity. The key value for each entity instance is computed in the same way as that for an observation instance.

The procedure of this algorithm is very straightforward. It processes the dataset in a row-wise manner. In dealing with each row, five steps are involved. The first step generates orphan measurement instances which are not connected to any observation instances. The second step groups these measurement instances according to their observation types. Then, for different observation types, the third and fourth steps materialize entity instances and observation instances respectively by either creating new oned or return existing ones. The last step assigns the context relationship among the different observation instances.

**Analysis of MaterializeDB**.

**Time**: As can be seen that this algorihm scans the original data file in a row-by-row manner without revisiting the already seen rows. So, it is linear in the size of the dataset. In addition, we use $EntIdx$ and $ObsIdx$ to facilitate the checking of unique entity and observation instances. Let $m$ be the number of distinct

keys, each checking could take $O(\log(m))$ time. So, in total, the algorithm runs in $O(n \log(m))$ where $m \ll n$ generally.

**Space**: $EntIdx$ and $ObsIdx$ are the intermdedite structures that we use in the algorithm. Since they keep the distinct entity and observation key values. The space complexity is of $O(m)$.

---

**Algorithm 1 MaterializeDB** $(Dataset, A.*)$

---

/* $Dataset$: [Input] in the form of a flat file */
/* $A$ : [Input] Annotations*/

$ObsIdx = \emptyset$; /* Keep index $\langle ObsTypeId, KeyVal \rangle \rightarrow ObsId$*/
$EntIdx = \emptyset$; /* Keep index $\langle ObsTypeId, KeyVal \rangle \rightarrow EntId$*/

**for** (each $Row\langle A_1, A_2, \cdots, A_n \rangle \in Dataset$) **do**
  /* Step 1: Define measurement instances */
  $MeasSet = \text{CrtMeasurement}(Row, A.*)$;

  /* Step 2: Partition the measurement instances according to observation types*/
  $ObsType2MeasIdx = \text{PartMeas}(MeasSet, A.*)$; /*$ObsType2MeasIdx = \{ObsTypeId \rightarrow \{mi\}\}$*/

  $ContextIdx = \emptyset$; /* Keep index $ObsTypeId \rightarrow ObsId$ to materialize context*/
  **for** (each $ObsTypeId$ in $ObsType2MeasIdx$) **do**
    /* Step 3: Find or create the entity instance for each observation type partition */
    $EntId = \text{MaterializeEntity}(ObsTypeId, ObsType2MeasIdx, EntIdx, A.*, OBOE.*)$;

    /* Step 4: Find or create the observation instance for each observation type partition of entity $EntId$ */
    $\text{MaterializeObs}(ObsTypeId, EntId, ObsType2MeasIdx, ObsIdx, ContextIdx, A.*, OBOE.*)$;
  **end for**

  /* Step 5: Assign the context observation instances */
  $\text{MaterializeContext}(ContextIdx, A.*, OBOE.*)$;
**end for**
**return** OBOE;

---

**Algorithm 2 CrtMeasurement** $(Row, A.*)$

---

/* Create new orphan measurement instances */
$MeasSet = \emptyset$; /* Keep the set of new measurement instances */
**for** (each $m = \langle MeasTypeId, ResAttribute, Cond, Val \rangle \in A.Map$) **do**
  **if**$((m.ResAttribute! = Row.A_i.Attrname)$ OR $(Row.A_i$ does not satisfy $m.Cond))$ **continue**;
  $mi_{id} = \text{GetNewMeasId}(OBOE.Measurement)$;
  **if** $(m.Val! = NULL)$ $MeasVal = m.Val$;
  **else** $MeasVal = Row.A_i.Val$;
  Create a measurement instance $\langle mi_{id}, null, MeasType, MeasVal \rangle$ and add it to $MeasSet$;
**end for**
**return** $MeasSet$;

---

---
**Algorithm 3 PartMeas** $(MeasSet, A.*)$

---
/* Partition measurement instances according to their observation types*/
$ObsType2MeasIdx = \emptyset$ /* Keep index for $ObsTypeId \rightarrow \{mi\}$ */
**for** (each $mi \in MeasSet$) **do**
$\quad ObsTypeId = $ GetObsTypeId $(A.MeasType, mi.MeasTypeId)$;
$\quad$ Update $ObsType2MeasIdx$ by changing the item $ObsTypeId \rightarrow \{mi\}$;
**end for**
**return** $ObsType2MeasIdx$;

---

---
**Algorithm 4 MaterializeEntity** $(ObsTypeId, ObsType2MeasIdx, EntIdx, A.*, OBOE.*)$

---
$KeyVal = $ GetObsTypeKeys $(ObsTypeId, ObsType2MeasIdx)$;
$HasKey = $ false;
**if** ($ObsTypeId$ has key measurements OR is specified with distinct yes) $HasKey = $ true;
$EntType = $ GetObsEntityType $(A.ObservationType, ObsTypeId)$;
$CrtNewEntInst = $ true;
**if** ($HasKey==$true) **then**
$\quad EntId = $ GetEntId$(ObsTypeId, KeyVal, EntIdx)$;
$\quad$ **if** ($EntId! = NULL$) $CrtNewEntInst = $ false;
**end if**
**if** ($CrtNewEntInst == true$) **then**
$\quad EntId = $ CrtEntId$(EntType)$;
$\quad$ Create an entity instance $ei = \langle EntId, EntType \rangle$ and put $ei$ to $OBOE.Entity$;
$\quad$ **if** ($HasKey==$true) $EntIdx = EntIdx \cup \{\langle ObsTypeId, KeyVal \rangle \rightarrow EntId\}$;
**end if**
**return** $EntId$;

---

| yr | spec | spp | dbh |
|------|------|------|------|
| 2007 | 1 | piru | 35.8 |
| 2008 | 1 | piru | 36.2 |
| 2008 | 2 | abba | 33.2 |

Table 2: Dataset 1

**Example 2.1 (Example with "key yes" and "distinct yes", without "identifying yes")** *Take the data in Table 2* [1] *as an example to explain the algorithm.*
*For* **Row(2007, 1, piru, 35.8)**

- *Step 1 creates four measurement instances:* $\langle mi_1, null, Year, 2007 \rangle$, $\langle mi_2, null, DBH, 35.8 \rangle$, $\langle mi_3, null, TaxonomicTypeName, Picea\ rubens \rangle$, $\langle mi_4, null, EntityName, 1 \rangle$, *and returns* $MeasSet = \{mi_1, mi_2, mi_3, mi_4\}$; [2].

- *Step 2 returns* $ObsType2MeasIdx = \{\{o_1 \rightarrow \{mi_1\}, o_2 \rightarrow \{mi_2, mi_3, mi_4\}\}$.

- *Step 3-4: for observation types* $o_1$ *and* $o_2$, *materialize entity and observation instance*

    - *for* $o_1$ *(with associated instance* $mi_1$ *of type* $m_1$*)*
        * *Since* $m_1$ *is specified as key, get the* $KeyVal = 2007$;
        * *No entity with this key exists in EntIdx, create an entity* $\langle ei_1, TemporalRange \rangle$; *Now,* $EntIdx = \{\langle o_1, 2007 \rangle \rightarrow ei_1\}$.

---
[1]The detailed annotation is in page 6 of Shawn's powerpoint file.
[2]For all the instances, the measurement characteristic is set to represent Measurement Type

---

**Algorithm 5 MaterializeObs**($ObsTypeId, EntId, ObsType2MeasIdx, ObsIdx, ContextIdx, A.*, OBOE.*$)

---

$KeyVal = \text{GetObsTypeKeys }(ObsTypeId, ObsType2MeasIdx)$;
$IsObsDistinct = \text{CheckIfObsDistinct}(A.ObservationType, ObsTypeId)$;
$CrtNewObsInst = \text{true}$;
**if** ($IsObsDistinct$==true) **then**
   $ObsId = \text{GetObsId}(ObsTypeId, KeyVal, ObsIdx)$;
   **if**($ObsId! = NULL$) $CrtNewObsInst = \text{false}$;
**end if**
**if** ($CrtNewObsInst == true$) **then**
   Create an observation instance $oi = \langle ObsId, EntId \rangle$ and put $oi$ to $OBOE.Observation$;
   **if** ($IsObsDistinct$==true) $ObsIdx = ObsIdx \cup \{\langle ObsTypeId, KeyVal \rangle \to ObsId\}$;
**end if**

/* Maintain the measurement instances for this observation instance */
$miSet = \text{GetMeasInst}(ObsType2MeasIdx, ObsTypeId)$;
**if** ($ObsId$ is a new one) **then**
   Set the $obsId$ to each $mi \in miSet$ so that $mi$-s are not orphans;
   Put all the $mi \in miSet$ to $OBOE.Measurement$;
**else**
   Discard all the $mi \in miSet$;
**end if**
$ContextIdx = ContextIdx \cup \{ObsTypeId \to ObsId\}$; /* $ContextIdx$ is also output*/

---

---

**Algorithm 6 MaterializeContext**($ContextIdx, A.*, OBOE.*$)

---

**for** ($ObsTypeId \to ObsId \in ContextIdx$) **do**
   $ContextObsTypeId, Rel = \text{GetContextObsTypeRel}(A.ContextType, ObsTypeId)$;
   **if** ($ContextObsTypeId! = NULL$) **then**
      $ContextObsId = \text{GetContextObsId}(ContextIdx, ContextObsTypeId)$;
      Create a context instance $ci = \langle ObsId, ContextObsId, Rel \rangle$;
      Put $ci$ to $OBOE.Context$;
   **end if**
**end for**

---

        * *Since $o_1$ is specified as* distinct, *need to make sure we do not create redundant observation instances. No entry with the key $\langle o_1, 2007 \rangle$ exists in ObsIdx, so, create an observation instance $oi_1$, which is of entity $ei_1$ and represented as $\langle oi_1, ei_1 \rangle$.*
           *Now, $ObsIdx = \{\langle o_1, 2007 \rangle \to oi_1\}$*
        * *Connect $mi_1$ to $oi_1$;*
      – *When deal with $o_2$,*
        * *$KeyVal = 1$.*
        * *Create an entity instance $\langle ei_2, Tree \rangle$; $EntIdx = \{\langle o_1, 2007 \rangle \to ei_1, \langle o_2, 1 \rangle \to ei_2\}$.*
        * *Create an observation instance $\langle oi_2, ei_2 \rangle$. No need to update ObsIdx because $o_2$ is not identified as* distinct.
        * *Connect $mi_2$, $mi_3$ and $mi_4$ to $oi_2$;*

• *Step 5 assigns the context relationship between $oi_1$ and $oi_2$;*

*For* **Row (2008, 1, piru, 36.2)**

• *Step 1 creates measurement instances $\langle mi_5, null, Year, 2008 \rangle$, $\langle mi_6, null, DBH, 36.2 \rangle$, $\langle mi_7, null, TaxonomicTypeName, Picea\ rubens \rangle$, $\langle mi_8, null, EntityName, 1 \rangle$ and returns $MeasSet = \{mi_5, mi_6, mi_7, mi_8\}$;*

5

- *Step 2 gets $ObsType2MeasIdx = \{\{o_1 \to \{mi_5\}, o_2 \to \{mi_6, mi_7, mi_8\}\}\}$*

- *Step 3-4: for observation types $o_1$ and $o_2$ materialize entity and observation instance*

    - *for $o_1$*
        * *$KeyVal = 2008$;*
        * *Create an entity instance $\langle ei_3, TemporalRange \rangle$;*
          *$EntIdx = \{\langle o_1, 2007 \rangle \to ei_1, \langle o_2, 1 \rangle \to ei_2, \langle o_1, 2008 \rangle \to ei_3\}$.*
        * *Create an observation instance $\langle oi_3, ei_3 \rangle$;*
          *$ObsIdx = \{\langle o_1, 2007 \rangle \to oi_1, \langle o_2, 1 \rangle \to oi_2, \langle o_1, 2008 \rangle \to oi_3\}$*
        * *Connect $mi_5$ to $oi_3$;*
    - *When deal with $o_2$,*
        * *$KeyVal = 1$.*
        * **item $\langle o_2, 1 \rangle \to ei_2$ is already in $EntIdx$, so get the entity id $ei_2$. No need to create an entity.**
        * *Since $o_2$ is not specified with* distinct *yet, we NEED to create an observation $\langle oi_4, ei_2 \rangle$. No need to update ObsIdx.*
        * *Connect $mi_6, mi_7, mi_8$ to $oi_4$;*

*For* **ROW (2008, 2, abba, 33.2)**

- *For $o_1$'s measurement 2008,*

    - *Since $\langle o_1, 2008 \rangle \to ei_3$ already exists in $EntIdx$,* **no need to create a new entity**.
    - *Since $o_1$ is specified with* distinct *yes, and $\langle o_1, 2008 \rangle \to oi_3$ already exists in ObsIdx,* **no need to create a new OBSERVATION** *and no need to put the measurement instance for 2008 into OBOE model.*

| plt | spp | dbh |
|-----|------|------|
| A | piru | 35.8 |
| A | piru | 36.2 |
| B | piru | 33.2 |

Table 3: Dataset 2

**Example 2.2 (Example with identifying)** *Let us use the data in Table 3 as an example.* [3] *Before we go through the algorithm step by step, we first note that $o_1$ and $o_2$ have key measurements m1 and $m_2$ respectively. So, we need to maintain the distinct entity instances for both of these two observation types. In addition, $o_1$ is specified with* distinct *yes while $o_2$ is not. So, we need to maintain the distinct observation instances for $o_1$ but not for $o_2$.*

*For the* **first row**,

- *The first step generates three measurement instances $MeasSet = \{\langle mi_1, null, EntityName, A \rangle,$*
  *$\langle mi_2, null, TaxonomicTypeName, Picea\ rubens \rangle, \langle mi_3, null, DBH, 35.8 \rangle\}$.*

- *The second step gets $ObsType2MeasIdx = \{\{o_1 \to \{mi_1\}, o_2 \to \{mi_2, mi_3\}\}\}$.*

- *For each observation type, create entity and observation instances.*

---

[3]The detailed annotation information is at page 8 in Shawn's powerpoint file.

– For $o_1$, the key value is $A$. Since there is no such a key in $EntIdx$, we create an entity $ei_1$ of type $Plot$.
$EntIdx = \{\langle o_1, A \rangle \rightarrow ei_1\}$.
We create an observation instance $oi_1$ whose entity is $ei_1$.
$ObsIdx = \{\langle o_1, A \rangle \rightarrow oi_1\}$.
Connect the measurement instance $mi_1$ to observation instances $oi_1$,

– For $o_2$, the key value is $(A, Picea\ rubens)$ since it has context $o_1$ with "identifying yes".
We create an entity instance $ei_2$ of type $Tree$.
$EntIdx = \{\langle o_1, A \rangle \rightarrow ei_1, \langle o_2, (A, Picea\ rubens) \rangle \rightarrow ei_2\}$.
We create an observation instance $oi_2$ whose entity is $ei_2$.
Connect the measurement instances $mi_2$ and $mi_3$ to observation instance $oi_2$.

- *The last step for this row is to connect the observations using context relationship. For this instance, we connect $oi_1$ to $oi_2$ with context "Within".*

*For the* **second row**,

- *The first step defines three measurement instances $MeasSet = \{\langle mi_4, null, EntityName, A \rangle, \langle mi_5, null, TaxonomicTypeName, Picea\ rubens \rangle, \langle mi_6, null, DBH, 36.2 \rangle\}$.*

- *The second step gets $ObsType2MeasIdx = \{\{o_1 \rightarrow \{mi_4\}, o_2 \rightarrow \{mi_5, mi_6\}\}\}$.*

- *For each observation type, create entity and observation instances.*

  – *For $o_1$, the key value is $\langle o_1, A \rangle$, $EntIdx$ already has an item for it with entity instance $ei_1$. No need to create a new instance for it.*
  *To create observation instance, since $o_1$ is specified with "distinct yes" and the key value is $\langle o_1, A \rangle$, which corresponds to an existing observatin instance $oi_1$. So, we do not need to create a new observation for it.*
  *When we try to connect the measurement instance $mi_4$ to observation instance, we realize that we did not create a new observation instance for type $o_1$. So its related measurement instance $mi_4$ can be discarded.*

  – *For $o_2$, the new key value is $\langle o_2, (A, Picea\ rubens) \rangle$, which corresponds to $ei_2$ in $EntIdx$, so no need to create a new instance for it either.*
  *To create observation instances, since no "distinct yes" is specified, we create a new observation instance $oi_3$ for it. Then, we connct the measurement instances $\{m_5, m_6\}$ to observation instance $oi_3$.*

*When we process the* **third row***, we have a new key value $\langle o_1, B \rangle$ for $o_1$, thus we create a new entity instance for it. For $o_2$, we have new key value $\langle o_2, (B,\ Picea\ rubens) \rangle$ and create a new entity instance for it. Similarly, we need to create new observation instances for both type.*