Keys and Conditional Key Discovery for Data Linking

Pablo Mollá Chárlez

February 8, 2025

Contents

1 Keys

A key is a set of properties that uniquely identifies every instance in the data.

1.1 Monotonicity & Minimality

- Key Monotonicity
  - **Definition:** If a set of properties K is a key, then any superset of K is also a key.
  - Explanation: Once a set of properties uniquely identifies every row (i.e., no two rows share the same values for all those properties), adding more properties cannot break that uniqueness—it can only maintain or further refine it.
- Minimal Key
  - **Definition:** A key is minimal if removing any one property from it causes it to stop being a key.
  - Explanation: This means no proper subset of that set of properties can serve as a key. Minimal keys are also sometimes called candidate keys because they are the smallest sets of properties that maintain uniqueness.

1.2 Multi-Valued Properties & Incomplete Descriptions

- Multi-Valued Properties
  - **Definition:** Properties that can take multiple values for a single entity (e.g., an actor who has performed in multiple movies).
  - Explanation: In order to compare these multiple values, either we proceed with value difference which refers to comparing individual values within a multi-valued property or, by set difference which typically refers to comparing the sets of values as a whole.
- **Incomplete Descriptions**
  - Optimistic Approach:
    - **Definition:** Any empty or missing value is considered different from all other existing values.
    - Implication: This maximizes the likelihood that a record with a missing value is still "unique" if all its other known values match another record's values.
  - Pessimistic Approach:
    - **Definition:** Any empty or missing value is considered possibly identical to existing values.
    - Implication: This minimizes the likelihood of treating a record with missing information as unique and instead assumes missing data might match existing known values elsewhere.

1.3 Types of Keys in Knowledge Graphs

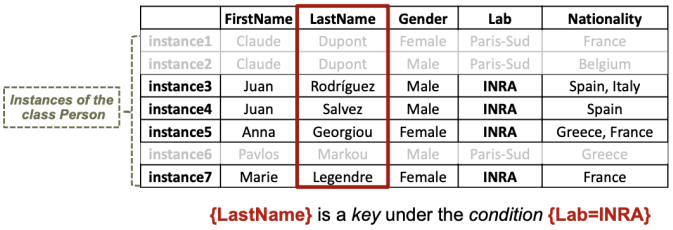
1. S-Key
   - **Definition:** A set of properties K is an S-Key if every row in the dataset has a unique combination of those properties under the following rules:
   - For set-valued properties, two sets are considered the same if they share at least one common element.
   - Empty values are considered different from existing ones (i.e., the optimistic approach to missing data).
2. F-Key:
   - **Definition:** A set of properties K is an F-Key: if every row in the dataset has a unique combination of those properties under the following rules:

- For set-valued properties, two sets are considered the same if they are exactly identical (i.e., have the same elements).
- Empty values are considered possibly identical to existing ones (i.e., the optimisitc approach to missing data).

3. SF-Key:
   - **Definition:** A set of properties K is an SF-Key if every row in the dataset has a unique combination of those properties under the following rules:
   - For set-valued properties, two sets are considered the same if they are exactly identical
   - Empty values are considered different from existing ones (i.e., the optimistic approach to missing data).

1.4 Conditional Keys

A conditional key is a set of properties that uniquely identifies all instances of a class only if those instances satisfy a certain condition (a set of property-value pairs). For instance, {LastName} can be a key for "Person" instances, but only if {Lab = INRA}. For people affiliated with "INRA," last name alone is sufficient to uniquely identify them.



	FirstName	LastName	Gender	Lab	Nationality
instance1	Claude	Dupont	Female	Paris-Sud	France
instance2	Claude	Dupont	Male	Paris-Sud	Belgium
instance3	Juan	Rodríguez	Male	INRA	Spain, Italy
instance4	Juan	Salvez	Male	INRA	Spain
instance5	Anna	Georgiou	Female	INRA	Greece, France
instance6	Pavlos	Markou	Male	Paris-Sud	Greece
instance7	Marie	Legendre	Female	INRA	France

Figure 1: Conditional Key

1.4.1 Quality Measures

We can distinguish mainly 2 measures:

- Support: The number of instances that both satisfy the condition (e.g., {Lab = INRA}) and instantiate (have values for) the key part $\implies$ Support = 4.
- Coverage: The ratio of that support to the total number of instances in the class. Formally, Coverage = $\frac{\text{Support}}{\#\text{AllInstances}}$. In the example would be Coverage = $\frac{4}{7}$.

These measures help ensure that the discovered conditional keys are both widely applicable (high coverage) and relevant to a significant subset of data (high support).

1.4.2 VICKEY: Mining Conditional Keys Efficiently

In section **Different Approaches: KD2R, SAKEY and VICKEY**, you can have a bit more of information about VICKEY.

- **Goal:** Given a dataset (with all instances of some class), a minimum support, and a minimum coverage, discover all minimal conditional keys whose support and coverage exceed those thresholds.
- **Challenge:** The search space can explode exponentially (roughly $\mathcal{O}(|V|^{|P|})$, where $|V|$ is the number of objects and $|P|$ is the number of properties).
- **Leveraging Non-keys:** Conditional keys can often be derived by starting from combinations that are known not to be keys (non-keys) and then adding property-value conditions that fix collisions.

- **Graph Exploration:** VICKEY systematically enumerates conditions.
    1. Start with single-property conditions $\{p = a\}$.
    2. Refine to two-property conditions $\{p_1 = a_1 \wedge p_2 = a_2\}$, etc.
    3. Continue until all minimal conditional keys are identified.

By using these steps and filtering out large parts of the search space via known non-keys, VICKEY manages to mine conditional keys efficiently, ensuring the discovered keys are both minimal and meet user-defined thresholds for support and coverage.

2 Key Discovery Approaches

2.1 SF-Keys: Keys and Pseudo-Keys Detection

2.1.1 Bottom-Up Approach

A **bottom-up approach** for discovering keys starts with checking single properties and moves on to pairs, triplets, etc., until it either finds a key or exhausts all property combinations. The rationale is:

1. **Single-property check:** Test if $P1$ alone is a key (i.e., it distinguishes all entities uniquely). If it is, then by key monotonicity, any superset containing $P1$ (like $P1P2$, $P1P3$, etc.) must also be a key.
2. **Increasing combination size:** If no single property is a key, proceed to all pairs of properties ($P1P2$, $P1P3$, etc.). Any pair that is found to be a key again triggers the same monotonicity principle, meaning all supersets of that pair (e.g., $P1P2P3$) will also be keys.
3. **Continue until you find the minimal keys (those whose proper subsets are not keys)**. If necessary, check triplets or even the entire set of properties.

2.1.2 Key Verification: Keys and Pseudo-Keys

To verify whether a set of properties K is truly a key, we typically:

- **Dataset Partition:** Partition the dataset based on the values of K. For each combination of values in K, gather all instances that share those values into a partition.
- **Check uniqueness:** if every partition has exactly one instance, then K is a key (no two entities have the same values for all properties in K).

For SF-pseudo keys (or almost keys), the same process applies, but you allow a small number of partitions to contain multiple instances - i.e., a few exceptions are tolerated. These pseudo keys can be valuable in real-world data settings where small inconsistencies or errors are common.

2.1.3 Quality Measures for (Pseudo-)Keys

Even if a set of properties is a key, it may only describe a fraction of the data or may not discriminate well in certain cases. Two common quality measures are:

1. Support:

$$\text{support}(P) = \frac{\#\,\text{instances described by } P}{\#\,\text{all instances}}$$

   This indicates the coverage of the property set—how many entities actually have values for those properties (versus missing data).

2. Discriminability:

$$\text{dis}(P) = \frac{\#\,\text{singleton partitions}}{\#\,\text{partitions}}$$

After creating partitions by grouping entities with the same values of P, discriminability checks the fraction of those partitions that are singletons (contain exactly one instance). A value of 1.0 would indicate a perfect key for the subset of data that actually has values for P.

Together, these measures help assess both how widely applicable a (pseudo-)key is (support) and how effectively it distinguishes instances (discriminability).

	Name	Actor	Director	ReleaseDate	Website	Language
film1	Ocean's 11	**B. Pitt** **J. Roberts**	S. Soderbergh	3/4/01	www.oceans11.com	---
film2	Ocean's 12	**B. Pitt** **J. Roberts**	S. Soderbergh R. Howard	2/5/04	www.oceans12.com	english
film3	Ocean's 13	B. Pitt G. Clooney	S. Soderbergh R. Howard	30/6/07	www.oceans13.com	english
film4	The descendants	N. Krause G. Clooney	A. Payne	15/9/11	---	english
film5	Bourne Identity	D. Liman	---	12/6/12	www.bourneIdentity.com	english

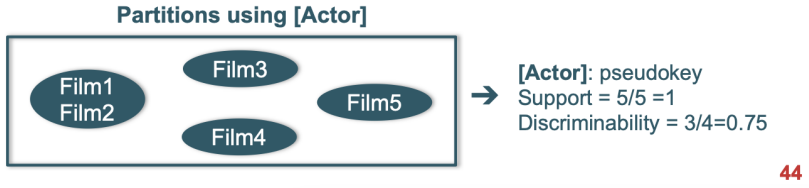


Figure 2: Support and Discriminality

In the example, support = $\frac{5}{5}$ = 1 means that every film (all 5) actually has Actors specified —so the "Actor" property covers all instances (100% coverage). If one film did not have any actor specified (empty value), then only 4 of the 5 films would be "described" by the **Actor property**, resulting in a support of 4/5. Besides, discriminability = $\frac{3}{4}$ = 0.75 comes from the fact that the "Actor" property creates 4 partitions overall ($\{film1, film2\}, \{film3\}, \{film4\}, \{film5\}$), and in 3 of those partitions there is only a single film (**film3**, **film4**, **film5**). Hence 3 singletons out of 4 total partitions yields 0.75.

2.2 F-Keys: Detection

2.2.1 Bottom-Up Approach

A bottom-up approach for F-keys works just like before: you first test each single property to see if it is an F-key (i.e., if it uniquely identifies every instance by exact-match of values); if not, you move on to property pairs, then triplets, and so on. Once a set of properties is found to be an F-key, key monotonicity guarantees all supersets are also F-keys.

2.2.2 Quality Measures

We distinguish 2 main quality measures for F-Keys:

- Discriminability(P): the number of instances that P can uniquely distinguish.
- Score(P) = $\frac{\text{Discriminability}(P)}{\#\text{instances}}$.
  - If Score(P) = 1, then P is a perfect key (it differentiates every instance).
  - If Score(P) < 1, then P is only a pseudo-key (some duplicates remain).

2.3 S-Keys: Detection

2.3.1 Key Discovery Is Complex

Among several reasons, key discovery is complex due to:

- **Exponential Search Space:** Checking all combinations of n properties means exploring up to 2^n subsets.
- **Data Scans:** For each combination, you need to verify uniqueness (or almost-uniqueness) by scanning every instance—costly for large or incomplete datasets.
- **Efficient Filtering & Pruning:** Techniques like key monotonicity (once a set is known to be a key, all its supersets are also keys) and non-key monotonicity (if a set is not a key, all its subsets are also not keys) **help cut down the search space**.

2.3.2 S-Keys vs. Other Types of Keys

Recall that an S-Key uses a "set-overlap" comparison for multi-valued properties and treats missing values optimistically (empty values are different from existing ones). This is often trickier than standard (F-Key) discovery because you need to handle partial overlaps of sets rather than exact matches.

2.3.3 Different Approaches: KD2R, SAKEY and VICKEY

- **KD2R (Pernelle et al.)** Focuses on data linking by discovering minimal keys automatically in RDF or similar structured data. It uses ontology mappings plus a bottom-up key discovery approach on each dataset (D_1, D_2). Only minimal keys discovered are then merged or compared to improve linkage between the two datasets.
- **SAKEY (Scalable Almost Key Discovery)** Addresses incomplete and erroneous data in large datasets. Discovers almost keys (a set of properties fails to be a key for only a small number of exceptions). The algorithm often starts by identifying non-keys (property sets that definitely do not provide uniqueness) and prunes them before searching for near-unique sets. This approach is well-suited for real-world RDF data with lots of missing or noisy values.

Exception of a key: an instance that shares values with another instance for a given set of properties *P*

- *f1, f2 and f3* are three exceptions for the property set {HasActor}

Exception Set E_P: set of exceptions for *P*

- E_P = {f1, f2, f3} ∪ {f2, f3, f4} = {f1, f2, f3, f4} for {HasActor}

Films	HasName	HasActor	HasDirector	ReleaseDate	HasWebsite	HasLanguage
f1	"Ocean's 11"	"B. Pitt" "J. Roberts"	"S. Soderbergh"	"3/4/01"	www.oceans11.com	---
f2	"Ocean's 12"	"B. Pitt" "G. Clooney" "J. Roberts"	"S. Soderbergh" "R. Howard"	"2/5/04"	www.oceans12.com	---
f3	"Ocean's 13"	"B. Pitt" "G. Clooney"	"S. Soderbergh" "R. Howard"	"30/6/07"	www.oceans13.com	---
f4	"The descendants"	"N. Krause" "G. Clooney"	"A. Payne"	"15/9/11"	www.descendants.com	"english"
f5	"Bourne Identity"	"D. Liman"	---	"12/6/12"	www.bourneIdentity.com	"english"
f6	"Ocean's 12"	---	"R. Howard"	"2/5/04"	---	---

Figure 3: Number of Exceptions

- **VICKEY** Another system for key discovery (or partial key discovery), typically employing advanced indexing or partitioning strategies to handle large-scale or heterogeneous data.

2.3.4 "Non-Key Discovery First" Strategy

Many algorithms begin by finding collisions - combinations of properties that definitely do not distinguish all instances. These identified "non-keys" prune large portions of the search space, because their supersets or subsets can be inferred quickly:

- Subsets of a non-key remain non-keys (they cannot suddenly become unique by dropping properties).
- Supersets might still be tested for uniqueness or near-uniqueness, but the presence of collisions guides the algorithm to refine or skip certain paths.
- Skip Irrelevant Combinations: If a property is incomplete (missing values for all or most instances) or clearly refers to another class (e.g., a property only used in a different domain), we can avoid exploring that property in your key-discovery search. Hence, combinations of properties that are obviously irrelevant—missing data, wrong class—need not be explored.
- Identify Potential n-non Keys When scanning the data, any set of properties that possibly yields $\geq n$ collisions is a candidate for being an n-non key.

An n-non key is simply a set of properties that fails to be unique for at least n instances (i.e., it has at least n "exceptions"). Formally, if $|EP|$ denotes the number of exceptions for property-set P, then P is an n-non key if $|EP| \geq n$. By identifying all the maximal n-non keys (the largest property-sets that still have $\geq n$ exceptions), one can deduce all the minimal $(n-1)$-almost keys, because those property-sets would now have fewer $(\leq n-1)$ exceptions. This technique helps prune the search space in almost-key discovery.

Given the following dataset, we can drop singletons to find the non-key properties:

Property	**Exceptions (before removing singletons)**
HasActor	$\{\{f1, f2\}, \{f1, f2, f3\}, \{f2, f3, f4\}, \{f4\}, \{f5\}\}$
HasDirector	$\{\{f1, f2, f3\}, \{f2, f3, f6\}, \{f4\}\}$
ReleaseDate	$\{\{f1\}, \{f2, f6\}, \{f3\}, \{f4\}, \{f5\}\}$
HasName	$\{\{f1\}, \{f2, f6\}, \{f3\}, \{f4\}, \{f5\}\}$
HasLanguage	$\{\{f4, f5\}\}$
HasWebsite	$\{\{f1\}, \{f2\}, \{f3\}, \{f4\}, \{f5\}, \{f6\}\}$

Table 1: Partitions after removing singletons

Property	**Exceptions (after removing singletons)**
HasActor	$\{\{f1, f2\}, \{f1, f2, f3\}, \{f2, f3, f4\}\} \implies$ Non-Key
HasDirector	$\{\{f1, f2, f3\}, \{f2, f3, f6\}\} \implies$ Non-Key
ReleaseDate	$\{\{f2, f6\}\} \implies$ Non-Key
HasName	$\{\{f2, f6\}\} \implies$ Non-Key
HasLanguage	$\{\{f4, f5\}\} \implies$ Non-Key
HasWebsite	$\{\} \implies$ Single Key

Table 2: Partitions after removing singletons

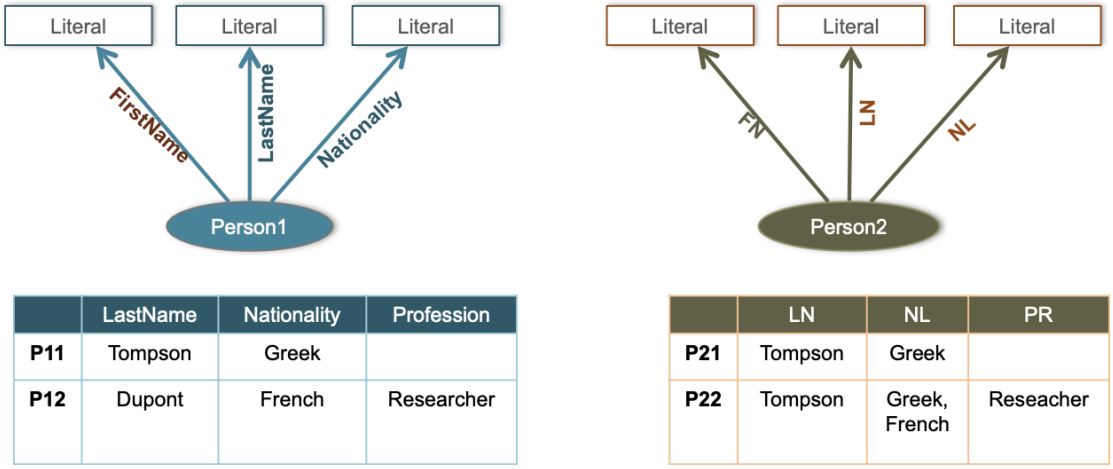
3 Key Limitations

- **No universal guarantee:** Some datasets simply have no keys at all.
- **Overly generic:** When keys exist, they usually apply to all instances of a class in the dataset, which might not reflect nuanced or local constraints.
- **Context dependence:** Real-world data often has conditions (e.g., "in German universities") under which a property set acts like a key, but not elsewhere.
- **Limited flexibility:** If a property is almost unique rather than fully unique, standard keys cannot capture this partial uniqueness.

4 LINKKEY: Data Interlinking

Data interlinking via LINKKEY is about discovering property-pair sets that uniquely link instances from two different classes—each class coming from different ontologies or datasets.

- **Setting:** We have two ontologies (**Ontology1**, **Ontology2**) and two corresponding datasets (Data of classA, Data of classB). Each dataset describes "similar" entities but using potentially different property names or schemas.
- **LinkKey Concept:** A LinkKey is a maximal set of property pairs $\langle p_1, p_2 \rangle$ such that, whenever two individuals (one from **classA**, one from **classB**) match on all those property pairs, they refer to the same real-world entity. "Maximal" means you cannot add more property pairs without losing that linking guarantee.
- **Example:** In **Dataset1** (Person1), you have LastName, Nationality, etc. In **Dataset2** (Person2), these might appear as LN, NL, etc. A discovered linkkey might be $\{\langle \text{LastName}, \text{LN} \rangle, \langle \text{Nationality}, \text{NL} \rangle\}$. If those properties match pairwise, it signals the same person across datasets.



	LastName	Nationality	Profession
P11	Tompson	Greek	
P12	Dupont	French	Researcher

	LN	NL	PR
P21	Tompson	Greek	
P22	Tompson	Greek, French	Reseacher

Figure 4: Linkkey

- **Role in Data Interlinking:** By automatically discovering LinkKeys, you can align (or "interlink") records across two datasets even when they use different vocabularies/ontologies. This forms the basis for entity reconciliation (finding duplicates) and data integration, enabling cross-dataset queries.

Hence, LinkKey discovery helps unify records from heterogeneous sources, ensuring that if two instances match on the LinkKey properties, they can be treated as the same entity.