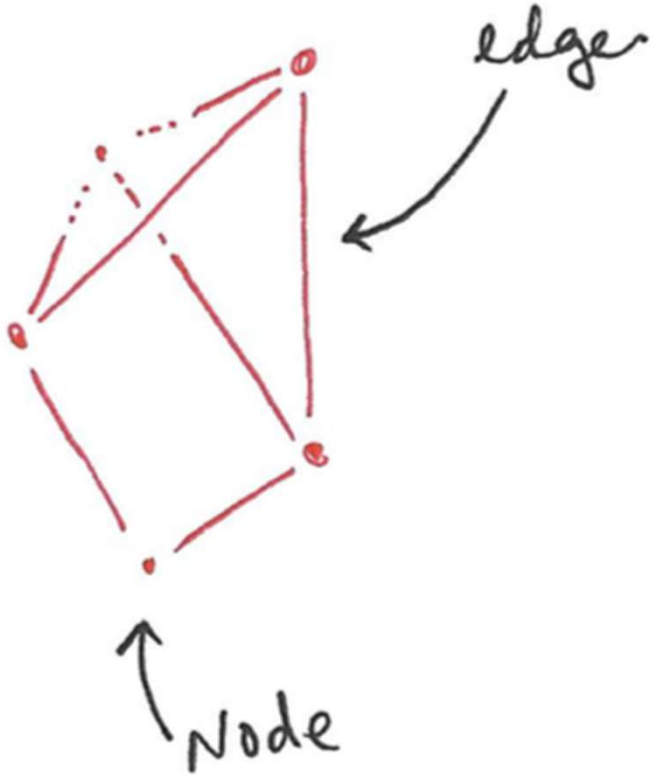
# Diseño y modelado en bases de datos de grafos



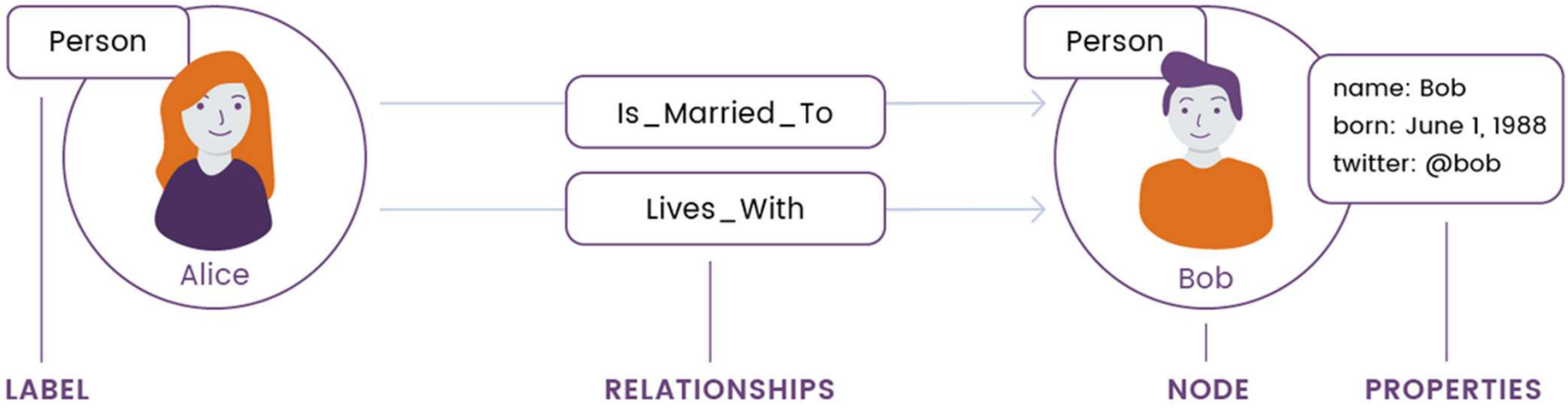

Instituto de Computación, FING, UdelaR

# Recordemos algunos conceptos

- No hay un único modelo lógico para bases de grafos
- Los modelos más populares actualmente son *property graph model (PGM)* y RDF + OWL

Nos concentraremos en PGM, luego veremos algunas ideas de diseño y reutilización de ontologías

# Elementos del PGM




Las etiquetas representan los tipos o roles de los nodos

Las propiedades son parejas (clave,valor) y pueden estar asociadas a nodos y a relaciones (aristas).

# El proceso de diseño

Consiste en decidir como representar a los elementos de la realidad en términos de nodos, etiquetas, relaciones y propiedades.

La realidad puede estar especificada en base a preguntas que quisiera poder contestar.

# Guías de diseño en PGM

# Guías de diseño (1)

Nodos: usualmente representan entidades de la realidad.

Se pueden identificar nodos para el modelo de grafos a partir de **sustantivos** del dominio (ej: una persona, una empresa, etc.)

Etiquetas: se usan para agrupar nodos en conjuntos.

Dan una noción “liviana” de tipos.

Se pueden definir restricciones sobre nodos con ciertas etiquetas.

Son opcionales, y cada nodo puede tener más de una etiqueta.

Fuente: https://neo4j.com/developer/guide-data-modeling/

# Guías de diseño (2)

**Relaciones:** vinculan entidades (nodos) y tienen una dirección.

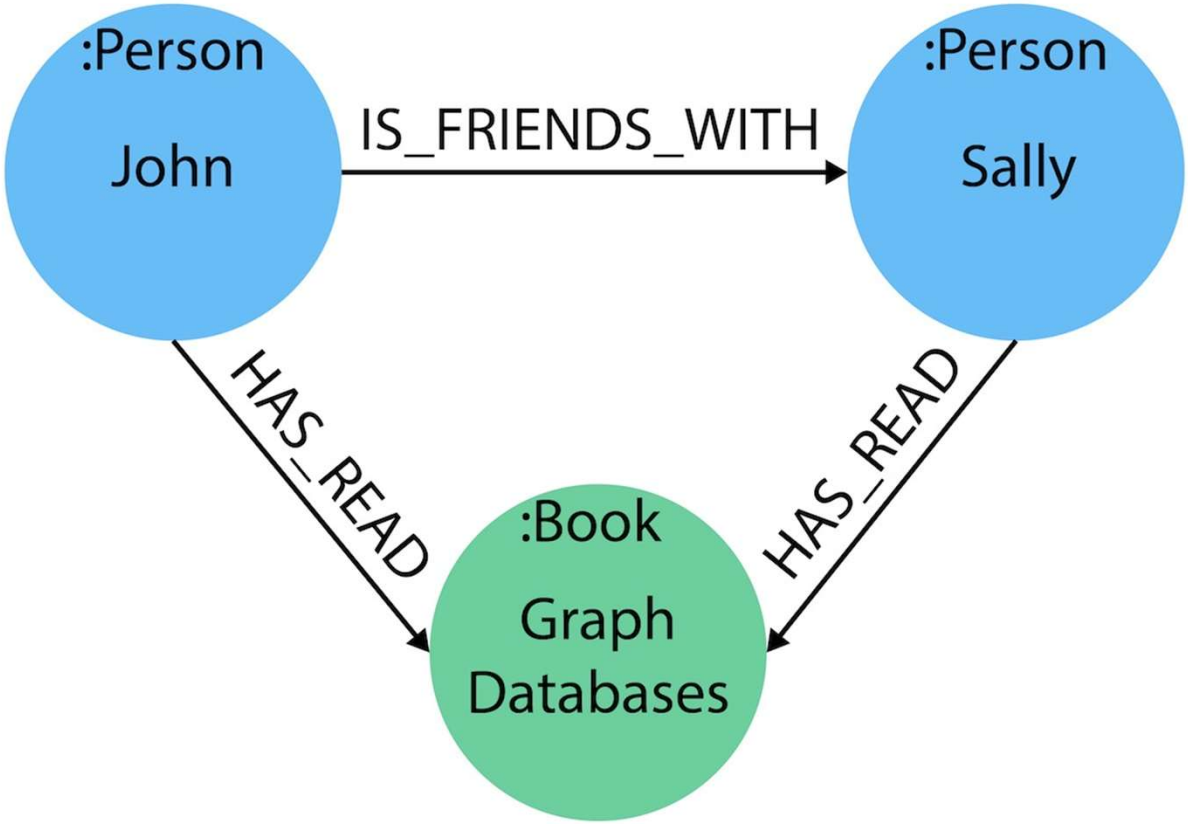
Se pueden identificar relaciones para el modelo de grafos a partir de verbos usados en el dominio (ej: dirige, conoce, lee, etc.)

**Propiedades:** permiten almacenar datos sobre las relaciones y los nodos.

Para identificarlas se puede partir de los atributos relevantes identificados en el dominio, y en las preguntas que se quieren responder

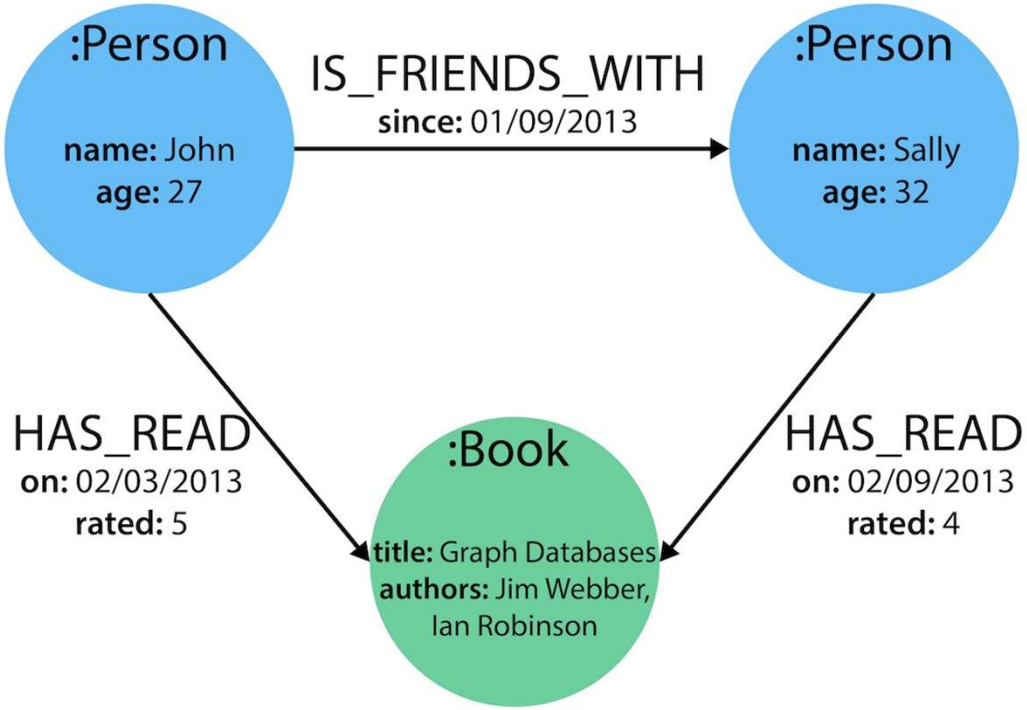
# Ejemplo a partir de una instancia

Dos personas, John y Sally, son amigos. Ambos leyeron el libro “Graph Databases”.



# Algunas preguntas posibles

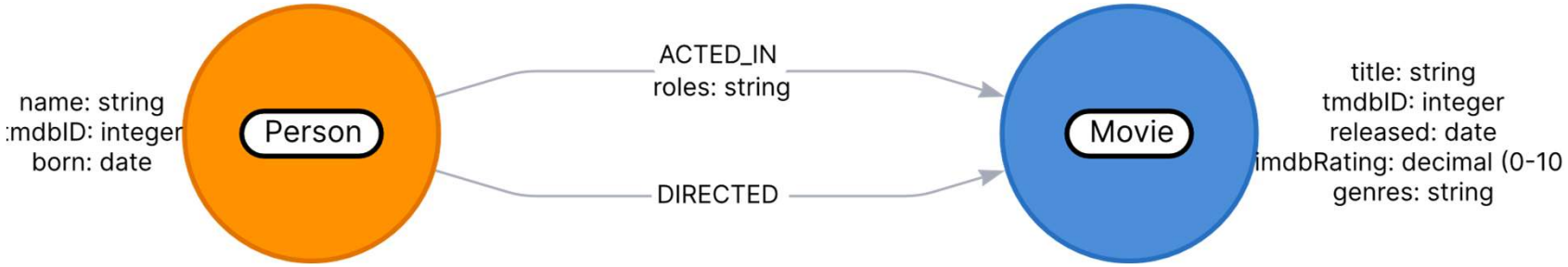
Cuánto hace que Sally y John son amigos?
Cuál es la calificación promedio del libro Graph Databases?
Quién es el autor del libro  Graph Databases?
Cuál es la edad de Sally? Y la de John?
Quién leyó el libro Graph Databases antes, Sally o John?



## Caso de ejemplo

- Movies example dataset, incluye películas, personas que actúan o dirigen películas y usuarios que califican películas.
- Interesa poder responder las siguientes preguntas:
  - ¿Qué personas actuaron en una película?
  - ¿Qué persona dirigió una película?
  - ¿En qué películas actuó una persona?
  - ¿Cuántos usuarios han valorado una película?
  - ¿Quién fue la persona más joven en actuar en una película?
  - ¿Qué papel interpretó una persona en una película?
  - ¿Cuál es la película mejor valorada en un año determinado según imDB?
  - ¿En qué películas dramáticas actuó un actor?
  - ¿Qué usuarios han puntuado una película con un 5?

Fuente: https://neo4j.com/developer/guide-data-modeling/




- Diferenciación entre una persona que actuó en una película, que dirigió una película y que calificó una película.
- Qué calificaciones se dieron, cuántas hay y cuándo se presentaron.
- Qué papel interpretó un actor en una película y cuál es su edad.
- Los géneros de las películas.

- **Representación de restricciones**

Neo4j ofrece varias restricciones para garantizar la calidad y la integridad de los datos en un grafo.

Restricciones de unicidad de propiedad: garantizan que los valores de propiedad combinados son únicos para todos los nodos con una etiqueta específica o para todas las relaciones con un tipo específico.

Sólo en la Edición Enterprise

Restricciones de existencia de propiedades: garantizan que una propiedad existe para todos los nodos con una etiqueta específica o para todas las relaciones con un tipo específico.

Restricciones de tipo de propiedad: garantizan que una propiedad tiene el tipo de propiedad requerido para todos los nodos con una etiqueta específica o para todas las relaciones con un tipo específico.

# Proceso de diseño

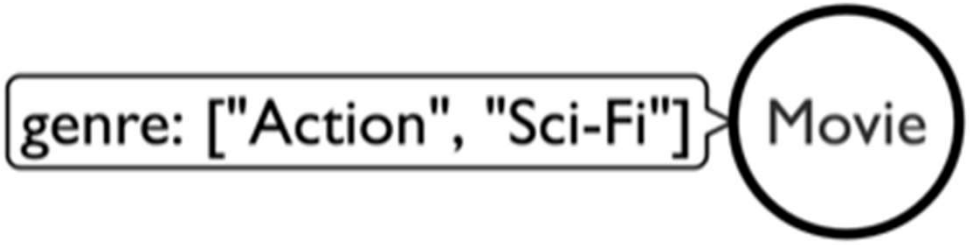
# Decisiones de diseño

Más allá de las recomendaciones generales algunos conceptos de la realidad tienen más de una representación posible.

¿Cuál elegir? Depende ...

Veremos a continuación algunas variantes y discusiones.

# Propiedades vs. Relaciones



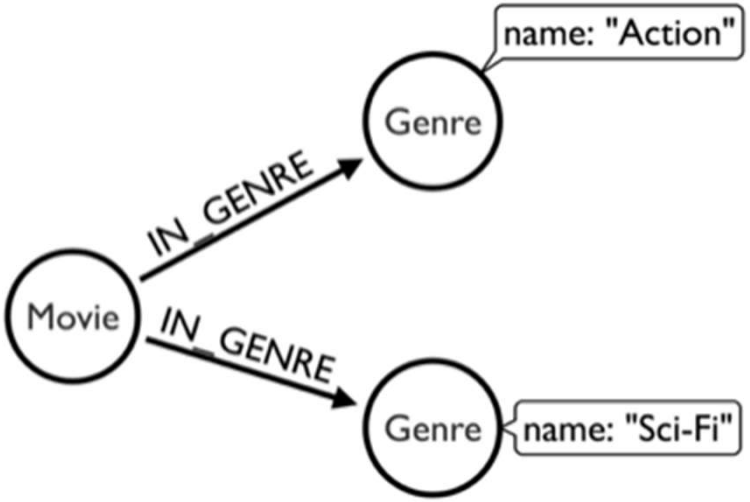

Devolver todos los géneros de una película

```
MATCH (m:Movie
{title:"The Matrix"})
RETURN m.genre;
```

Devolver todas las películas que comparten géneros

```
MATCH (m1:Movie), (m2:Movie)
WHERE any(x IN m1.genre WHERE x IN m2.genre)
AND m1 <> m2
RETURN m1, m2;
```

# Propiedades vs. Relaciones (2)




Devolver todos los géneros de una película

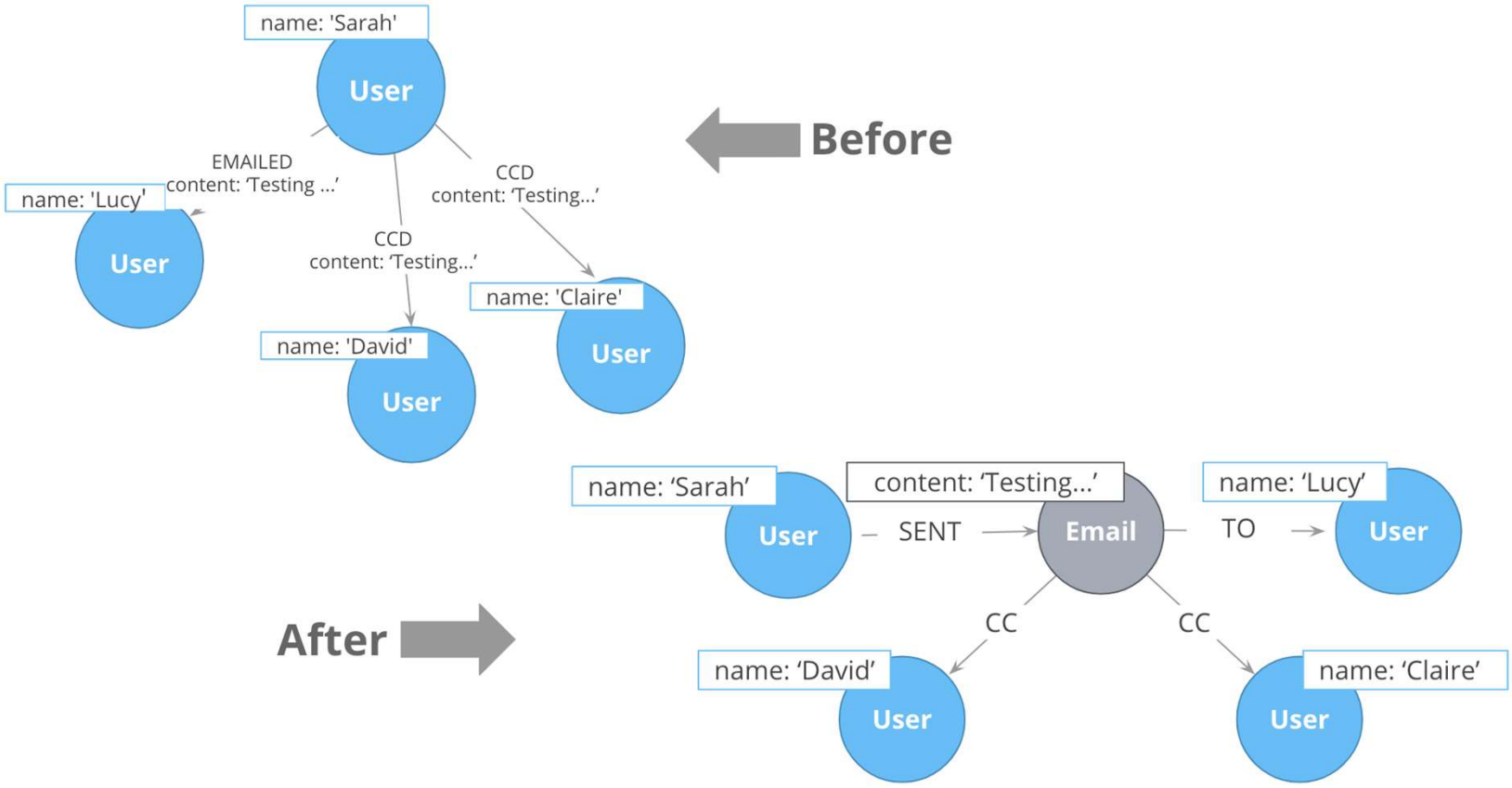
```
MATCH (m:Movie {title:"The Matrix"}),
      (m)-[:IN_GENRE]->(g:Genre)
RETURN g.name;
```

Devolver todas las películas que comparten géneros

```
MATCH (m1:Movie)-[:IN_GENRE]->(g:Genre),
      (m2:Movie)-[:IN_GENRE]->(g)
RETURN m1, m2, g
```

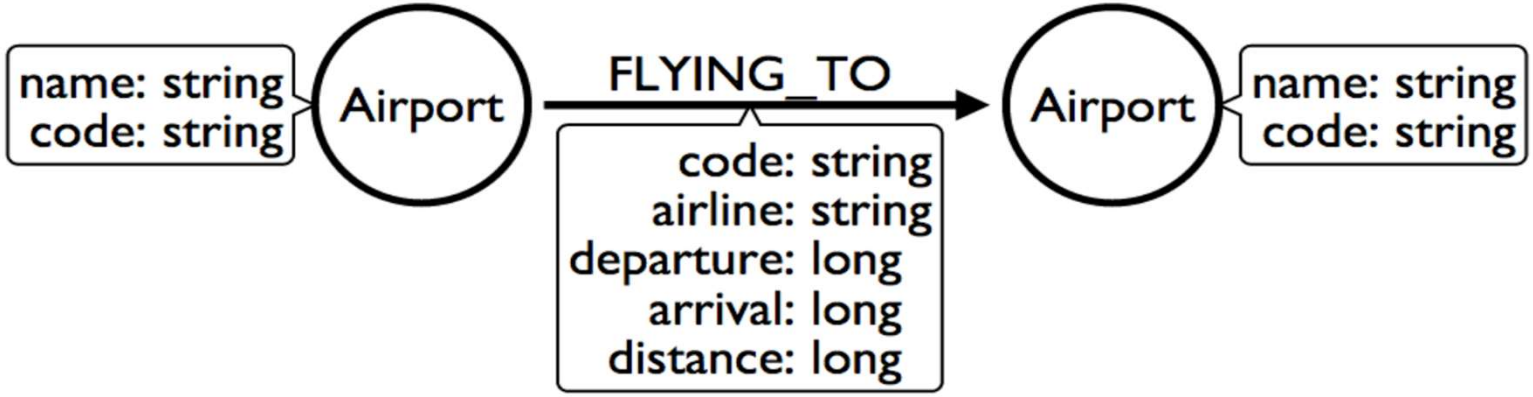
# Hiper aristas o nodos intermedios

Permiten representar relaciones entre más de dos entidades



Comenzamos con un modelo muy simple.

Supongamos que se quiere usar el grafo en una aplicación para búsqueda y reserva de vuelos.

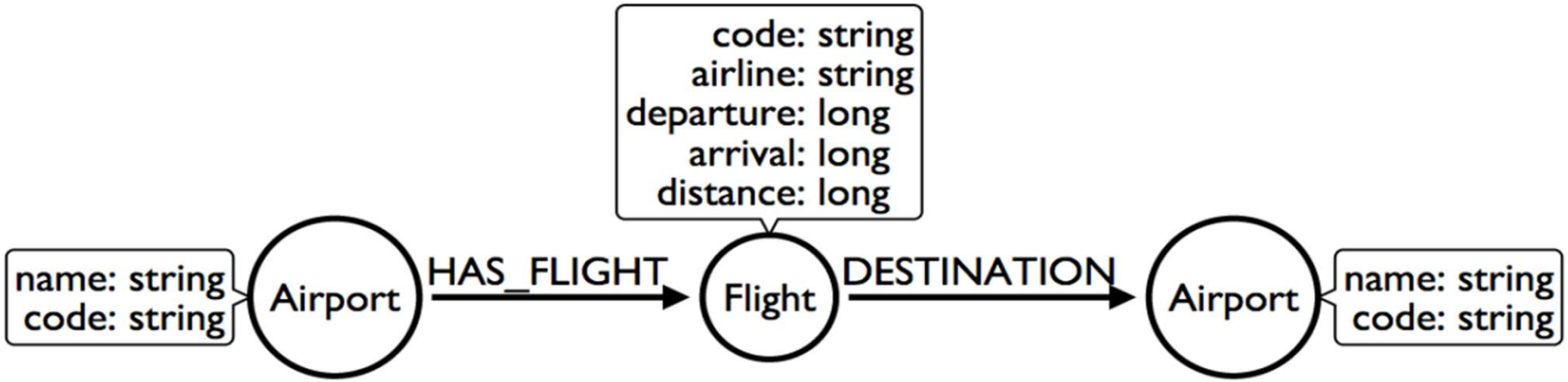



El vuelo como relación no parece ser una buena idea

Fuente: https://maxdemarzi.com/2015/08/26/modeling-airline-flights-in-neo4j/
https://www.slideshare.net/slideshow/neo4j-training-modeling/235729895

# Modelado y refinamiento (2)

Ahora el vuelo es un nodo




No es un mal modelo, pero algunos nodos tendrán muchísimas aristas entrantes o salientes (ej: aeropuertos como Pekín, Dubai, Londres Heathrow).

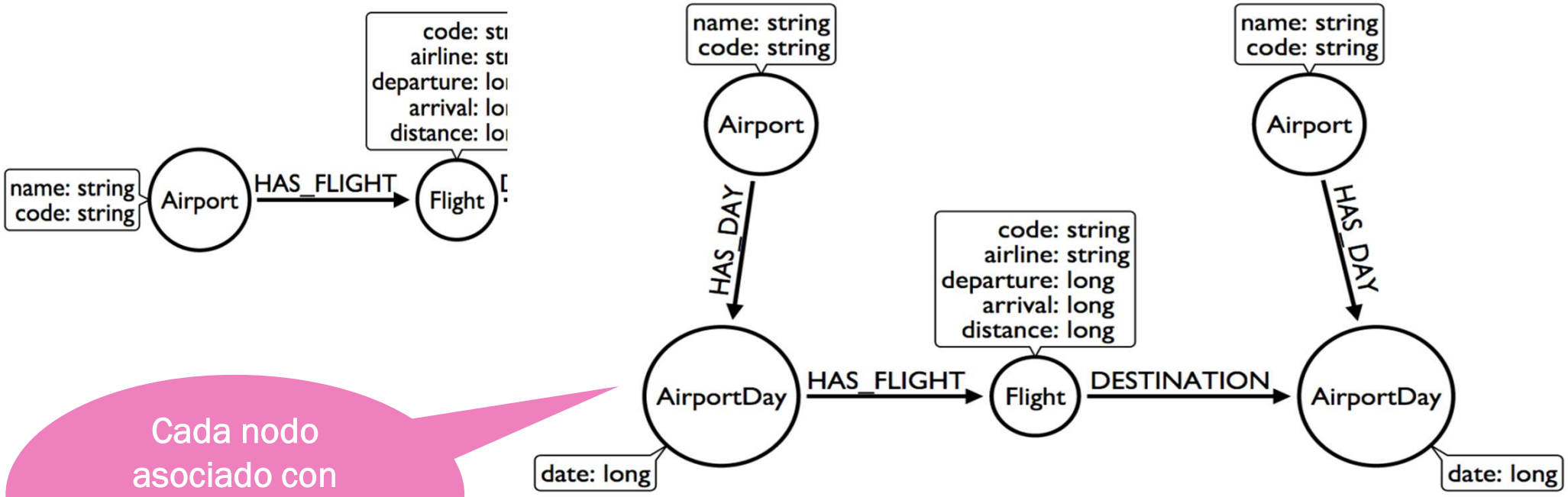
Para filtrar rutas hay que chequear múltiples propiedades

Reduce el desempeño!

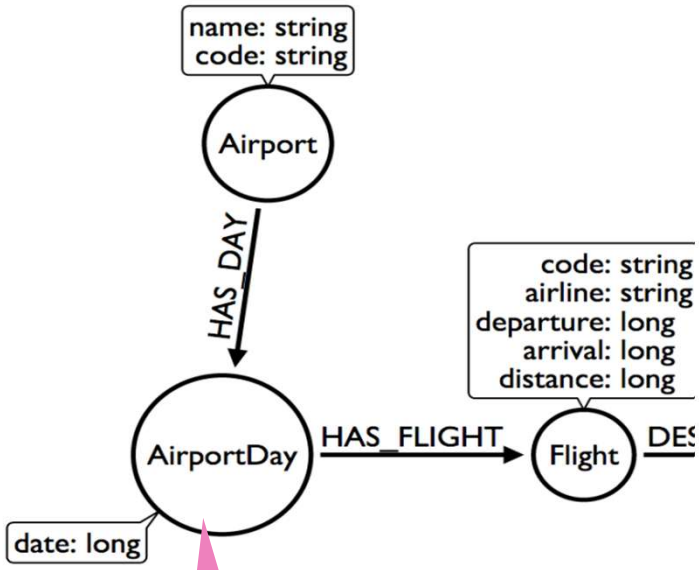
# Modelado y refinamiento (3)

OBJETIVO: poder reducir rápidamente el subgrafo en que hago las búsquedas

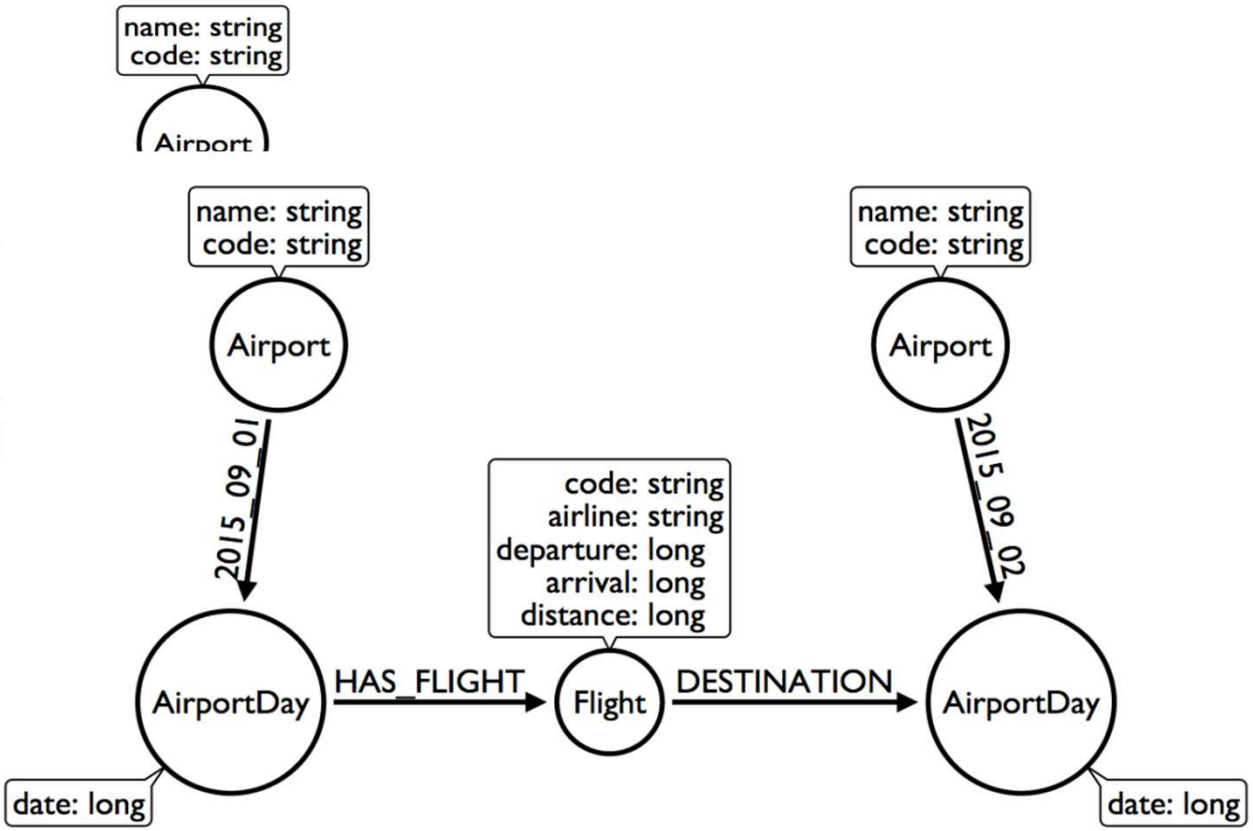
Tener en cuenta las consultas: ejemplo, para buscar un vuelo sabemos el origen y el destino que interesa, y también la fecha del vuelo.



# Modelado y refinamiento (4)

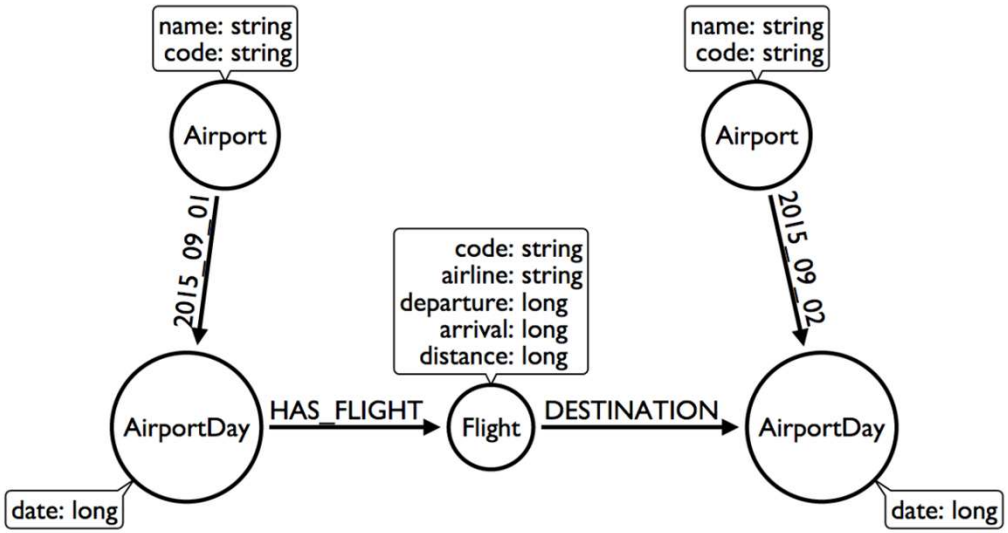



Filtrar los nodos en cierta fecha requiere filtro en la propiedad date



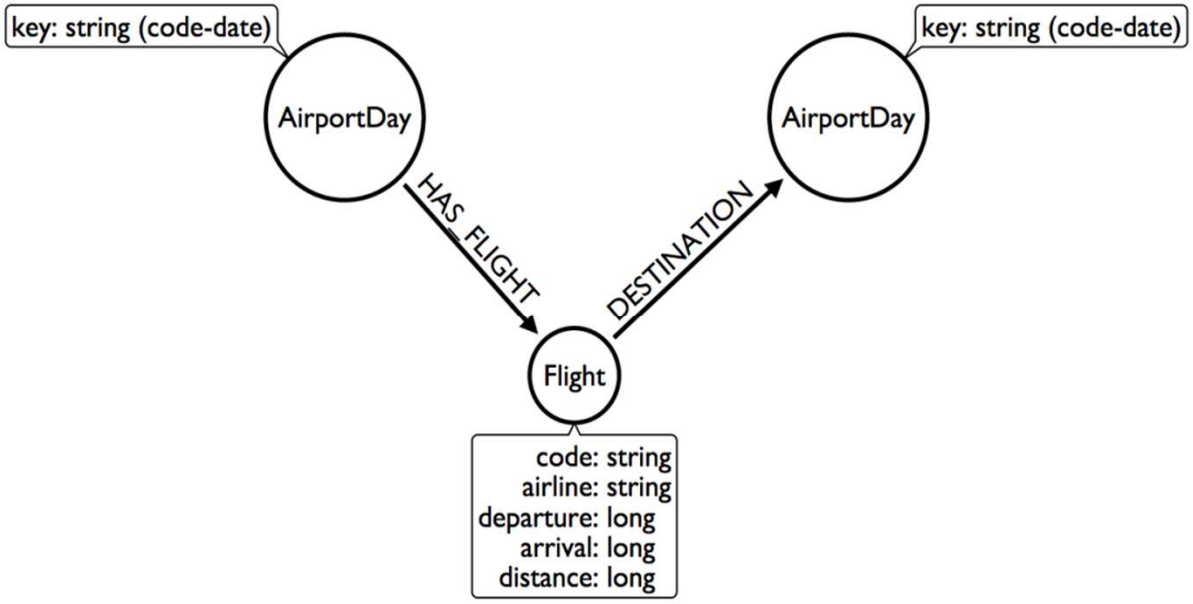

Usamos la fecha como una relación. Navego desde un nodo Aeropuerto al nodo *AirportDate* por cierta relación sin tener que comprobar la propiedad fecha de 365 relaciones.

# Modelado y refinamiento (5)




Genero una nueva clave combinando el código del aeropuerto y la fecha (epoch)

¿Qué nos falta para acceder al *AirportDate* directamente?



- Además de las metodologías de modelado industriales que hemos visto, hay propuestas académicas que hacen foco en:
  - Modelado conceptual de las bases de datos de grafos
  - Representación del esquema y de las restricciones (PG-Schema)


Pokorný, J. (2016, July). Conceptual and database modelling of graph databases. In *Proceedings of the 20th international database engineering & applications symposium* (pp. 370-377).

Angles, R., Bonifati, A., Dumbrava, S., Fletcher, G., Green, A., Hidders, J., ... & Zivkovic, D. (2023). Pg-schema: Schemas for property graphs. *Proceedings of the ACM on Management of Data*, *1*(2), 1-25.

Skavantzos, P., & Link, S. (2025). Entity/Relationship Graphs: Principled Design, Modeling, and Data Integrity Management of Graph Databases. *Proceedings of the ACM on Management of Data*, *3*(1), 1-26.

# Migrando datos desde otros modelos

- **Migrando datos relacionales**: Vamos a considerar solo la estrategia de mapeo directo:
  - Entidades a nodos
  - Relaciones binarias a aristas
  - Nodos intermedios para relaciones de 3 o más

  No es la única estrategia posible.

  Ver Knowledge Graphs book, sección 6.4

# Diseño de ontologías en RDF y OWL

- Ingeniería de ontologías: desarrollo y aplicación de metodologías para la construcción de ontologías, proponiendo procesos para construir y mantener ontologías de mejor calidad.
- Las primeras metodologías se basaban en un proceso tipo cascada, en el que los requisitos y la conceptualización se fijaban al comienzo [Grüninger y Fox, 1995a, Fernández et al., 1997, Noy y McGuinness, 2001].
  - Herramientas: [Gómez-Pérez et al., 2006, Keet, 2018, Kendall y McGuinness, 2019].
- Para situaciones que implican ontologías grandes o en constante evolución, se han propuesto formas más iterativas y ágiles de construir y mantener ontologías.

- Las metodologías ágiles modernas incluyen :
  - eXtreme Design (XD) [Presutti et al., 2009, Blomqvist et al., 2016]
  - Simplified Agile Methodology for Ontology Development (SAMOD) [Peroni, 2016]
  - Modular Ontology Modelling (MOM) [Shimizu, 2023]

- Estas metodologías suelen incluir dos elementos clave:
  - requerimientos ontológicos y
  - la aplicación de patrones de diseño ontológico

Lectura recomendada!

[Shimizu , 2023] Shimizu, C., Hammar, K., & Hitzler, P. (2023). Modular ontology modeling. *Semantic Web*, *14*(3), 459-489.

- Requerimientos ontológicos
- Especifican la tarea que se pretende realizar con la ontología o el grafo de conocimiento.

Suelen expresarse a través de preguntas de competencia (CQ) [Grüninger y Fox, 1995b], que son preguntas en lenguaje natural que ilustran las necesidades de información típicas a las que uno necesitaría que respondiera la ontología o el grafo de conocimiento.

Las CQ pueden complementarse con restricciones adicionales y requisitos de razonamiento.

TESTING: Pueden usarse las CQ implementadas como consultas sobre un conjunto de datos de prueba y asegurarse de que los resultados esperados se cumplen.