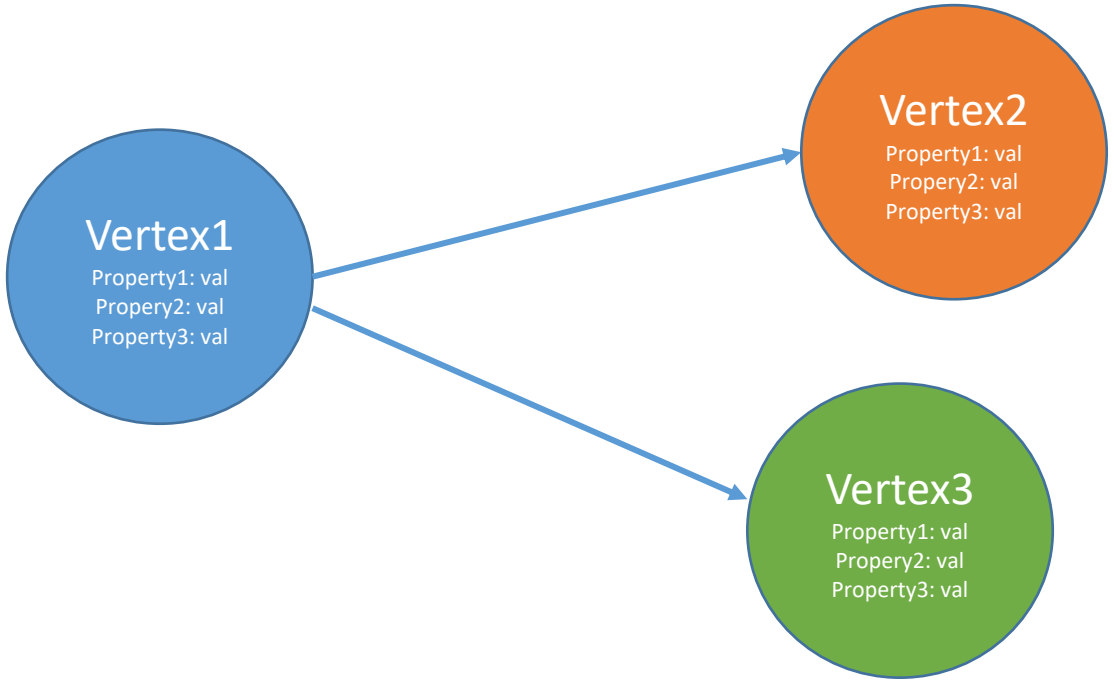
Modeling Breaking Changes in AAI

Jimmy Forsyth, AT&T, AAI PTL
Chandra Cinthala, AT&T

2 April 2019

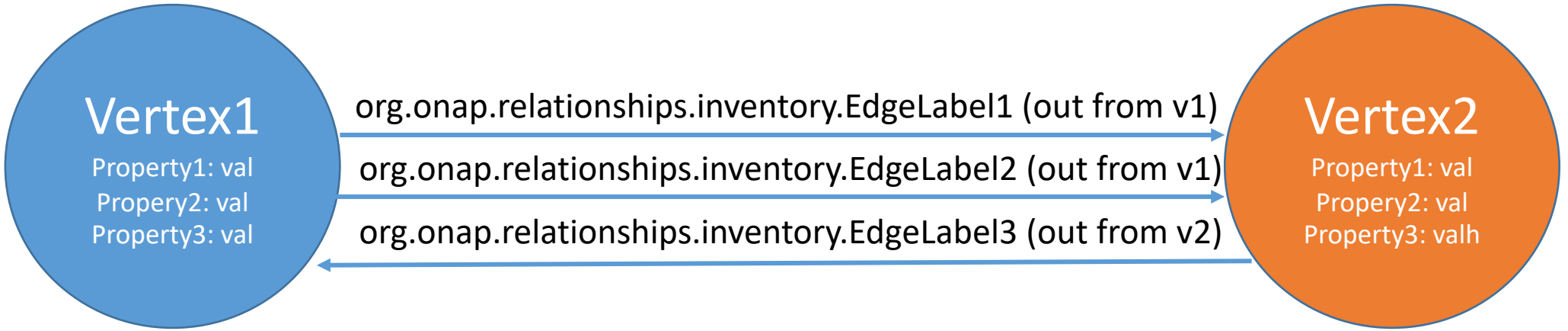
AAI Graph Basics

- A&AI uses Janusgraph for persistence which is a property graph model, where a graph is a set of vertices with edges between them.
- JanusGraph stores graphs in adjacency list format which means that a graph is stored as a collection of vertices with their adjacency list.



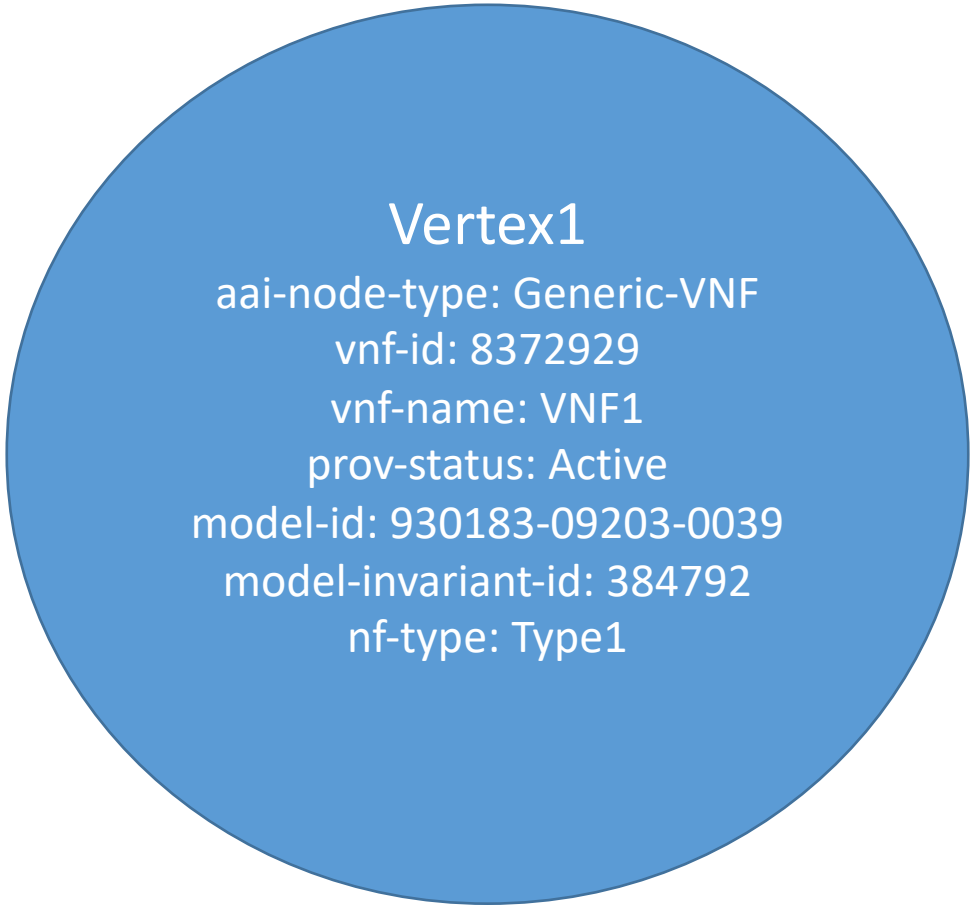
AAI Graph Basics

- The adjacency list of a vertex contains all of the vertex's incident edges (and properties).



AAI Graph Basics

- A vertex is the fundamental unit of the graph and represents an object. Vertex can have properties to describe the object.



AAI Graph Basics

- An edge is a connection between two vertices that expresses a relationship between them. An edge can have a multiplicity, direction, and properties.

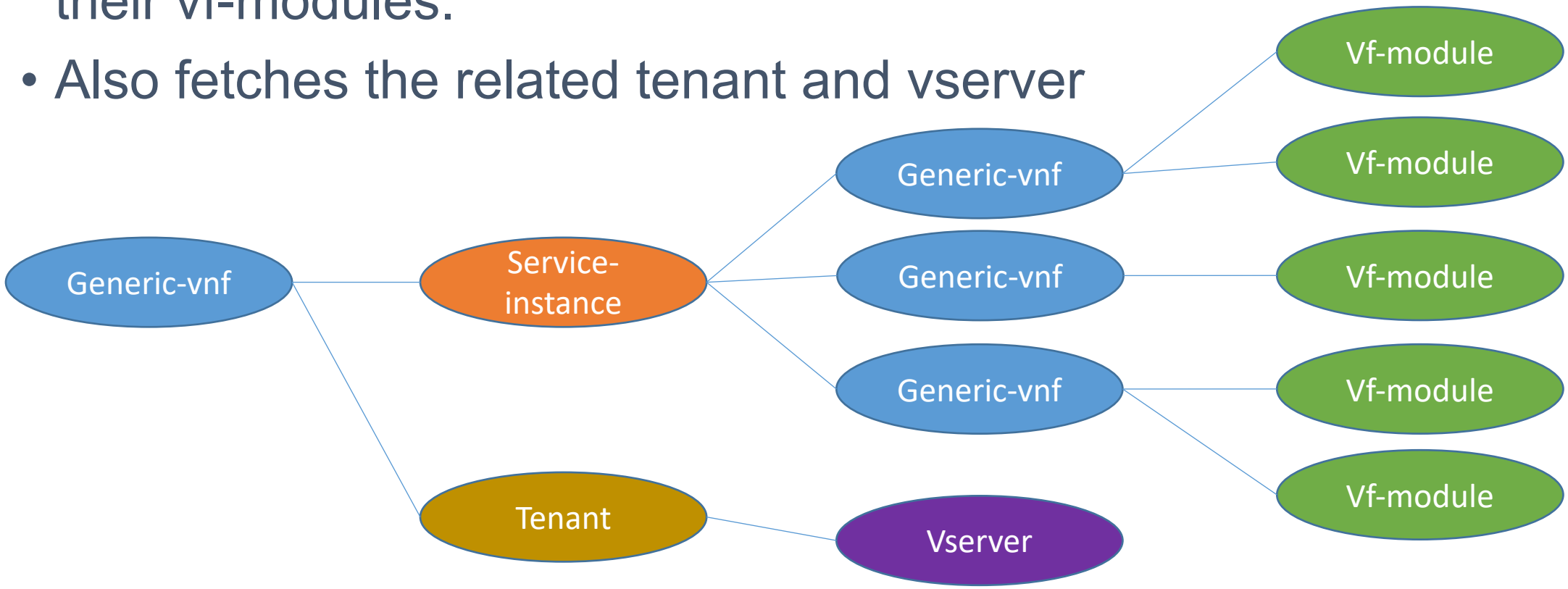
```
{
        "from": "vf-module",
        "to": "generic-vnf",
        "label": "org.onap.relationships.inventory.BelongsTo",
        "direction": "OUT",
        "multiplicity": "MANY2ONE",
        "contains-other-v": "!${direction}",
        "delete-other-v": "!${direction}",
        "prevent-delete": "NONE",
        "default": "true",
        "description":""
},
```

AAI Graph Basics

- Traversal is the process of analyzing a graph's structure.
- Traversals discover and returns information about edges, vertices, and their properties.
- In graph databases, the relationship is a primary component of the data model and traversing from vertex to edge to vertex and beyond is the primary mechanism for querying the data within the model.

AAI Graph Basics

- Traversal Example: A custom query that starts at a generic-vnf, follows the edge to the service-instance, then gets all connected vnfs and their vf-modules.
- Also fetches the related tenant and vserver



AAI Model Change Guidelines

- AAI tries to provide backward compatibility between versions of the API
- Extra properties are ignored
- Changes tend to be additive
- In Dublin, AAI is providing support for v11 (Amsterdam), v12, v13 (Beijing), v14 (Casablanca), v15, and v16 (Dublin)
- v12 and v15 are ECOMP contributions and can be used by clients, but we document the official ONAP versions on the wiki

AAI Model Change Guidelines

- Schema Change Guidelines:
- Design for query ability: The query patterns drive the graph data model
  - Model Vertices for entities
  - Node Properties
    - Represent entity attributes, metadata
  - Model Edges for structure
    - Express connections between entities/vertices
    - Establish semantic context for each entity
  - Relationship Properties
    - Express strength, weight or quality of the relationship or metadata
- Properties vs Relationship
  Use relationships when you need to establish semantic context/connection between vertices AND/OR the property value comprises of a complex value type.
  Example: Find all pservers in cloud-region/zone (complex value type)

AAI Model Change Guidelines

- Use properties when there's no need to qualify the relationship AND the attribute value comprises a simple value type
  Example: Find all pservers of same equip-type.
  Small property lookup on a node will be a less expensive operation than traversing a relationship. However, many small properties on a node, or a lookup on a large string or large array property will impact performance.
- Try to use UUID property as the unique identifier for all vertices
- A Traversal Query is always better than using many GETs to traverse multiple nodes.

AAI Model Change Guidelines – Things to AVOID

- Modeling entities as relationships
- Lots of attribute-like properties
- Property value redundancy
- Adding arbitrary properties to a vertex but rather use vertex and edge
- Using vertex property as a foreign key but rather use relationship/edge to the other vertex.
- Don’t overload a property
  - E.g.: using single “type” property for representing type/role/function of the node.

Case Study: PNF for 5G Plug and Play Use Case

- The PNF object is an example of an object that violates our own rules
  "Try to use UUID property as the unique identifier for all vertices"
- The change to PNF was proposed in Dublin https://wiki.onap.org/display/DW/Proposal+to+Change+AAI+PNF+Entity+to+use+PNF-ID+as+key
- PNFs were initially stored in pserver
- Pserver has hostname as the key (likely a mistake)
- PNF was cloned from pserver, so pnf-name used as key (definitely a mistake)
- The fact that pnf-id is described as a UUID really intends for the pnf-id to have a unique id within the context of an ONAP instance, although future use cases might lead us to want it to be universally unique

Case Study: PNF for 5G Plug and Play Use Case

- Manage PNFs and VNFs as alike as possible, i.e., like network functions
- Default the SourceName in the ONAP Aware Software on the PNF to [VVV][Serial#] but permit the operator to set it to a value following their own naming conventions (just as they would do with a VNF name).
- Support PNF plug and play flow
- Establish future direction for operators to be able to use network functions in topologies without specifying whether they are PNFs or VNFs (let the orchestrator use, e.g., policy to resolve)

Case Study: PNF for 5G Plug and Play Use Case

- When this change was proposed, no operators were using PNF objects in their deployments
- The time to do this change would have been Casablanca but the proposal came too late to get concurrence from all projects for a breaking change
- During the Casablanca/Dublin timeframe, the PNF object started being used by carriers – migration strategy needed!
- The JanusGraph schema is always built using the latest version of the AAI data model – key change is a breaking change for all versions of the API since it can't be backward compatible

Aligning the AAI Data Model

- There are other object types that are “incorrectly” modelled in AAI, according to the guidelines
- Other Examples of non-uuid keys:
  - dvs-switch uses switch-name
  - instance-group uses id instead of instance-group-id
  - l-interface, lag-interface, and p-interface all use interface-name
  - lag-link uses link-name
  - network-profile uses nm-profile-name
  - physical-link uses link-name
  - platform uses platform-name
  - project uses project-name
  - pserver uses hostname
  - service-subscription uses service-type

Handling Breaking Changes in AAI

- The pnf-name to pnf-id key change highlights the issue of making changes on objects that have data in existing deployments
- Care in aligning the AAI data model up-front will save grief down the road
- AAI is socializing the PNF key change in the Dublin release notes so providers can attempt to mitigate the need for migration
- The modeling subcommittee's recommendations should be informed by the existing AAI data model
- AAI is undertaking to produce a UML model in papyrus based on the AAI data model, but existing tooling is unable to process the enormous AAI REST API
- Should the ONAP community have a global strategy for handling these kinds of migrations?