Investigation: Custom Keyed Collections

Databases

Variety of syntaxes for composite keys. Triggers often propagate using triggers or cascading. Keys are not hashed.

Computed keys (custom identity) can be used as primary keys but not in all databases, and not in certain scenarios with dynamic behavior. General workflow around custom/computed identity is to use multiple tables and cascade between them.

Ruby

Custom hash keys are tied to instances. No way to perform customization per collection.

compare_by_identity

Forces a Hash to use identity rather than custom hash keys.

object.object_id

Returns an integer that is unique for the lifetime of an object.

Python

Uses integer types for identity as hash codes. No way to perform customization per collection. Emulating container types does not strictly bind the implementation of types to a specific abstract base. This works because using collections is done by separate functions such as slice() and not methods of collections.

__eq__ / __hash__

Requires that objects be considered equal if hashes are equal.

id

Returns an integer that is unique for the lifetime of an object.

Java

Uses integer types for identity as hash codes.

Object.equals / Object.hashCode

Allows objects to override the alter the default identity via hash codes and equality for keys but requires that objects with identity to be equal, otherwise collections might not work properly.

IdentityHashMap

Does not extend standard collection type but implements interface to Map to create class with differing identity and equality for storing keys.

java.System.identityHashCode

Helper to avoid virtual identity and get runtime referential identity. String allocations are not considered equal if pointing to different references.

CSharp

Uses integer types for identity as hash codes..

IEqualityComparer

Requires implementing a way to generate identity using hash codes and compare values.

Object.GetHashCode / Object.Equals

Allows objects to override the alter the default identity via hash codes and equality for keys but requires that objects with identity to be equal, otherwise collections might not work properly.

RuntimeHelpers.GetHashCode

Helper to avoid virtual identity and get runtime referential identity. String allocations are not considered equal if pointing to different references.

Rust

Uses integer types for identity as hash codes. No known way to generate a unique id via the system, but can compare pointers and use Rc<T> to create identity based functionality.

std.HashMap.with_hasher

Allows passing in a function to customize hashing for keys, requires Eq and Hash be implemented for the keys such that.

trait Hash / trait Eq

Allows objects to override the alter the default identity via hash codes and equality for keys but requires that objects with identity to be equal, otherwise collections might not work properly.

Observations

Overriding behaviors

Languages allow customization of the concept of equality per type or object. JavaScript does not currently allow overriding equality operations. These languages also allow overriding of identity via hashing.

JS does not allow overriding equality nor customizing identity currently. It seems unlikely that existing referential identity semantics of maps would be allowed to change if equality cannot also be customized by looking at requirements of implementing custom identity and equality in other languages.

Subclassing

Very few cases encouraging subclassing exist, custom types and wrapped types such as PersonByEmail, PersonByFamily, PersonByLastName, etc. seem to be the path of least friction to provide different views of identity for a single Person object.

Hooks into existing collections

Databases, Rust, and C# allow for providing custom computations into the identity of a collection. These require the non-strictly enforced semantics of identity and equality in both Rust and C# which can lead to logical errors or misuse if the implementations of those 2 traits are not correctly synchronized. Databases do not allow for differentiation of equality and identity.