1 Vysvětlivky v dokumentu:

- *element značí, že daný element je reference
- zápis $g_1 - g_3$ znamená guardy indexované od 1 do 3, přijde mi to jako nejjednodušší značení
- K [Column] značí kolekci elementů typu Column
- precedessor(i) je třída nacházející se o i úrovní výše v stromu dědičnosti, tzn. dědí od ní třída, na které je testován guard
  Matematicky vyjádřeno: precedessor(1) = parent
  precedessor(i) = precedessor(i - 1).parent
  $i \in \mathbb{N}$
- $\mathbb{N}$ je zápis množiny přirozených čísel
- root - index kořenu hierarchického stromu - nemá předka

2 Primitivni typy

- $guard : isPrimitive = true$
- $\omega_1(\pi_0, guard)$
- $\pi_0 : class \rightarrow K[column]$
- $class \in Class, column \in TableColumn$
- Třída, která je primitivní se nenamapuje do databázového modelu na Tabulku. Tyto tridy se namapuji na Sloupce v databázi daného typu.

3 EmbeddedClass

-
- $guard_1 : isPrimitive = true$
- $\omega_1(\pi_1, guard1)$
- $\pi_1 : class \rightarrow column$
- $class \in Class, column \in TableColumn$
- Embedded třídy se namapují na sloupce třídy, na níž jsou embedded

4 Obecné guardy

Obecné guardy by měly splňovat všechny třídy, které jsou serializovatelné, jde o to, aby nebyly Primitive a měly právě jedno ID a nebyly embedded

- $g_1 : isPrimitive = false \wedge isEmbedded = false$
- $g_2 : \exists i \in \mathbb{N} : properties[i].isID = true$

5 Třídy bez serializovatelného předka

- $g_3 : parent = NULL$
- $\omega_1(\pi_2, g_1 - g_3)$
- $\pi_2 : class \rightarrow \{table, K[column], primaryKey\}$
- $class \in Class, column \in TableColumn, table \in Table, primaryKey \in PrimaryKey$
- mapovací pravidlo namapuje třídu na nově vzniklou tabulku, množinu jejich sloupců a primární klíč

6 Třídy s jednoduchou dědičností

Všechny typy dědičnosti předpokládají již namapovanou třídu předka, není nutný předpoklad namapování všech předchů v hierarchickém stromě, tento je rekurzivně splněn, pokud je namapován předek, matematicky znězorněno: $g_4 : parent.resolve(Table) = true$

6.1 Joined nebo implicitní inteheritanceType

- $g_5 : inheritanceType = \_\_not\_defined" \wedge parent \neq NULL \forall j \in \mathbb{N} : j \leq root : precedessor(j).inheritanceType = "\_\_not\_defined"$
- $g_6 : (parent \neq NULL) \wedge ((inheritanceType = parent.InheritanceType \wedge inheritanceType = Joined) \vee inheritanceType = NULL \wedge \exists i \in \mathbb{N} : \forall j \in \mathbb{N} : j < i : precedessor(j).inheritanceType = NULL, precedessor(i).inheritanceType = Joined)$
- $\omega_3(\pi_3, g_1 \wedge g_2 \wedge g_4 \wedge (g_5 \vee g_6))$
- $\pi_3 : class \rightarrow \{table, K[column], *parentPrimaryKeyColumn, K[precedessorColumn], foreign$
- $class \in Class, column \in TableColumn, table \in Table, primaryKey \in PrimaryKey, foreignKeyinForeignKey, parentPrimaryKeyColumn, parentColumn \in TableColumn$

- typ Joined vytvoří novou tabulku, sloupce a referuje na id sloupec předka
- data se budou vkládat do všech sloupců tabulek předků označované jako $K[precedessorColumn]$

6.2 TablePerClass

- $g_7 : (parent \neq NULL) \wedge ((inheritanceType = parent.InheritanceType \wedge inheritanceType = TablePerClass) \vee inheritanceType = NULL \wedge \exists i \in \mathbb{N} : \forall j \in \mathbb{N} : j < i : precedessor(j).inheritanceType = NULL, precedessor(i).inheritance TablePerClass)$
- $\omega_5(\pi_4, g_1 \wedge g_2 \wedge g_4 \wedge g_7)$
- $\pi_4 : class \rightarrow \{table, K[Column], primaryKey\}$
- $class \in Class, column \in TableColumn, table \in Table, primaryKey \in PrimaryKey$

6.3 SingleTable

- $g_8 : (parent \neq NULL) \wedge ((inheritanceType = parentInheritanceType \wedge inheritanceType = SingleTable) \vee inheritanceType = NULL \wedge \exists i \in \mathbb{N} : \forall j \in \mathbb{N} : j < i : precedessor(j).inheritanceType = NULL, precedessor(i).inheritanceType SingleTable)$
- $\omega_4(\pi_3, g_1 \wedge g_2 \wedge g_4 \wedge g_8)$
- $\pi_3 : class \rightarrow \{*table, K[column], *K[treeColumn] * primaryKey\}$
- $class \in Class, column \in TableColumn, table \in Table, primaryKey \in PrimaryKey, treeColumn \in TableColumn$
- Typ SingleTable referencuje tabulku, primární klíč, vytvoří nové sloupce vstupní třídy, součástí namapované tabulky jsou sloupce celé hierarchie
- Na datové vrstě budou data vkládána i do sloupců hierarchické struktury v zápisu označovaných jako K[treeColumn]

7 Přechodové třídy mezi různými inheritanceTypy - nedodělané

7.1 sth to SingleTable

- $g_9 : parent \neq NULL \; \exists i \in N : \forall j \in \mathbb{N} j < i : precedessor(j).inheritanceType = NULL, (precedessor(i).inheritanceType = TablePerClass \vee precedessor(i).inheritanceType = Joined), self.inheritanceType = SingleTable$

- $\omega_6(\pi_4, g_1 - g_2 \wedge g_9)$
- $\pi_3 : E \to \{*table, K[Column], *primaryKey\}$

7.1.1 sth to Joined

- $g_(10) : parent \neq NULL \; \exists i \in \mathbb{N} : \forall j \in \mathbb{N} j < i : precedessor(j).inheritanceType = NULL, (precedessor(i).inheritanceType = TablePerClass \vee precedessor(i).inheritanceType = Joined), self.inheritanceType = SingleTable$
- $\omega_5(\pi_4, g_1 - g_2 \wedge g_1 0)$
- $\pi_4 : E \to \{table, K[Column], FK\}$