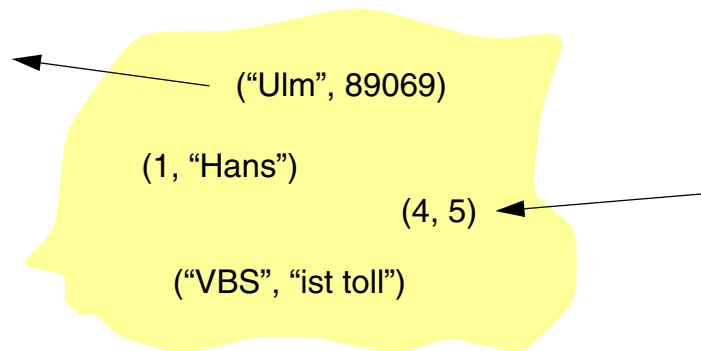


## 2.10 Tupelräume

---

- Linda Tuple Space
  - ◆ ursprüngliche Idee von David Gelernter, 1985
  - ◆ verteilter Dienst speichert Tupelmengen
  - ◆ Komponenten der Tupel enthalten Werte



## 2.10 Tupelräume (2)

---

- Operationen
  - ◆ read
    - finden eines geeigneten Tupels im Tupelraum durch Matching (Werte bestimmter Komponenten müssen gleich sein)
    - z.B. ("Hans", "Müller", \*, \*) matcht ("Hans", "Müller", "TI", 1.0)
    - Rückgabe eines gefundenen Tupels
  - ◆ take
    - wie read jedoch wird das Tupel vom Tupelraum entfernt
  - ◆ write
    - ein Tupel wird im Raum abgelegt
    - es darf mehrere Tupel mit gleichen Werten geben



## 2.10 Tupelräume (3)

---

- **Operationssemantik**
  - ◆ atomare Operationen
  - ◆ Operationen erscheinen streng sequentiell selbst bei nebenläufigen Zugriffen
  
- ★ **Vorteil**
  - ◆ Verknüpfung von Kommunikation und Koordinierung erleichtert Anwendungsentwicklung
    - z.B. take wartet bis entsprechendes write kommt (Erzeuger-Verbraucher-Koordinierung)



## 2.10 Tupelräume (4)

---

- ★ **Einsatzgebiete**
  - ◆ Implementierung verteilter Algorithmen
  - ◆ Grundlage für einfache Kommunikation zwischen Anwendungskomponenten
  
- **Weiterentwicklung: Java Spaces**



## 2.11 Java-Spaces

---

- Teil der Jini-Architektur
  - ◆ Dienste für Java-Spaces erscheinen als Jini-Dienste
  
- Architektur von Linda mit Erweiterungen
  - ◆ Einträge (Tupel) werden durch Java-Objekte gebildet
    - enthalten auch Operationen (Methoden)
    - **aber:** wie bei Linda enthält Space nur Objektkopie
  - ◆ Komponenten sind voll typisiert (Java-Typen und -Klassen)
  - ◆ Polymorphismus (Matching von Subklassen bei Suche nach Einträgen mit Typ der Basisklasse)
  - ◆ Transaktionssemantik über mehrere Operationen hinweg möglich
    - Auswirkungen sind entweder alle (commit) oder gar nicht (abort) sichtbar

