taran tool

in-memory база данных и сервер приложений

In-memory

- Все хранимые данные должны помещаться в оперативную память

План

- Схема хранения
- Индексирование

История развития

История развития

- in-memory cache

История развития

- in-memory cache
- in-memory persistent cache

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db
- in-memory multiindex db

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db
- in-memory multiindex db
- in-memory multiindex db with lua functions

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db
- in-memory multiindex db
- in-memory multiindex db with lua functions
- in-memory multiindex db with lua cooperative runtime

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db
- in-memory multiindex db
- in-memory multiindex db with lua functions
- in-memory multiindex db with lua cooperative runtime
- in-memory multiindex db with lua app server

История развития

- in-memory cache
- in-memory persistent cache
- in-memory persistent cache with replication
- in-memory key/value db
- in-memory multiindex db
- in-memory multiindex db with lua functions
- in-memory multiindex db with lua cooperative runtime
- in-memory multiindex db with lua app server
- hybrid storage multiindex db with sql and lua app server

Схема данных

- Данные храняться в спейсах, это аналог таблиц
- В спейсах храняться таплы, это аналог строк в таблицах
- Значения данных находятся в полях таплов, это аналог значения в колонке

Схема данных

- Например, тапл

```
{
  «В чем ответ на вопрос о мироздании?»,
  42
}
```

- Таплы могут хранить вложенные структуры

```
{
  «В чем ответ на вопрос о мироздании?»,
  {
      author = 'Дуглас Адамс',
      text = 42,
  }
}
```

Схема данных

- Спейс можно также назвать массивом таплов
- По умолчанию схема данных в Tarantool нестрогая
- Схема данных может быть ограничена тем, на каких полях какого типа строятся индексы

Схема данных

- Для более строгой схемы можно применить простой набор правил, в котором можно перечислить имена полей и их типы
- В Tarantool нет понятия баз данных, как в Redis или MongoDB. Можно сказать, что Tarantool — одна база данных

Язык доступа

Для доступа к данным из коннекторов используются простые команды:

- insert
- get
- update
- delete
- select
- replace

Для более сложных запросов используются хранимые процедуры на языке Lua.

Операции доступа к данным
Создание спейса (таблицы)

```
box.schema.space.create('theanswers', {if_not_exists=true})
```

Типы данных
Атомарные

- строковый
- числовой
- с плавающей точкой
- с десятичной точкой
- uuid

Комплексные

- объект
- массив

Задание схемы для спейса

```
box.schema.theanswers:format(
  {
    {name="question", type="string"},
    {name="answer",   type="string"}})
```

- Задание или изменение схемы в любой момент
- Если есть данные в спейсе, то они должны соответствовать задаваемому формату
- Каждая вставка объекта будет валидироваться по указанной схеме

Индексы

- Может быть много
- Из нескольких полей (композитные)
- Типы индексов
  - tree (B+*) — древовидный
  - hash — хешиндекс
  - bitmap — битовый
  - rtree — пространственный
  - functional — на основе функции
  - json path — по json пути внутри документа

Создание индекса

- Самый первый — первичный

```
box.space.theanswers:create_index('primary',
  {
    parts = {
      {field="question"}},
  type='TREE',
  if_not_exists=true})
```

- Другие — вторичные

```
box.space.theanswers:create_index('answer',
  {
    parts = {
      {field="answer"},
      {field="question"}},
  type='TREE',
  if_not_exists=true})
```

Локальные операции с данными — insert

```
box.space.theanswers:insert(
  {"В чем ответ на вопрос о мироздании?", "42"})
```

- Если тапл уже был, то операция вернёт ошибку

get — поиск одного тапла

```
box.space.theanswers:get({"В чем ответ на вопрос о мироздании?"})
```

update — обновление одного или нескольких таплов

```
box.space.theanswers:update(
  {"В чем ответ на вопрос о мироздании?"},
  {
    {'=', 'answer', "43"}})
```

- Если в каком-то уникальном индексе уже есть такое значение, то ошибка

delete — удаление тапла

```
box.space.theanswers:delete("В чем ответ на вопрос о мироздании?")
```

- Если тапла не существует, ошибки не будет

replace — замена тапла целиком

```
box.space.theanswers:replace(
  {"В чем ответ на вопрос о мироздании?", "42"})
```

- Если тапл уже был, то он будет полностью перезаписан

pairs — итерация по индексу

```
for _, tuple in box.space.theanswers:pairs() do
    print(tuple)
end
```

- Итерация по индексу
- Можно по возрастанию или по убыванию
- На ходу направление менять невозможно
- Можно выполнять дополнительные проверки
- Можно прервать итерацию в любой момент

select — получение данных

```
box.space.theanswers:select("В чем ответ на вопрос о мироздании?")
```

- Если не указать условие, Tarantool попытается вернуть **все данные**, будьте осторожны!

В заключение

- Tarantool — in-memory хранилище с гибкой схемой данных
- Индексирование одного или нескольких полей
- Произвольное количество индексов
- Итерация по индексам