

ets, dets



Co to toto

Biblioteki (moduły):

- ETS - Erlang (built-in) Term Storage
- DETS - Disc based version of ETS

Ale po co toto?

Ale po co toto?



Wydajność

ets

- zbiór tabel
- wbudowany, uruchamiany wraz z węzłem Erlanga
- tabela ściśle związana z procesem

ets - cechy

- szybkość, niezawodność (atomowość i izolacja)
- prostota obsługi
- może symulować działanie bazy danych

ets - przegląd funkcji

`ets:new(Name, [Option]).`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

Type = `ordered_set` | `set` | `bag` | `duplicate_bag`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

Type = `ordered_set` | `set` | `bag` | `duplicate_bag`

Access = `private` | `protected` | `public`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

Type = `ordered_set` | `set` | `bag` | `duplicate_bag`

Access = `private` | `protected` | `public`

`named_table`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

Type = `ordered_set` | `set` | `bag` | `duplicate_bag`

Access = `private` | `protected` | `public`

`named_table`

`{keypos, Position}` albo `{keypos, #RecordName.FieldName}`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

Type = `ordered_set` | `set` | `bag` | `duplicate_bag`

Access = `private` | `protected` | `public`

`named_table`

`{keypos, Position}` albo `{keypos, #RecordName.FieldName}`

`{heir, Pid, Data}` | `{heir, none}`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

`Type = ordered_set | set | bag | duplicate_bag`

`Access = private | protected | public`

`named_table`

`{keypos, Position}` albo `{keypos, #RecordName.FieldName}`

`{heir, Pid, Data} | {heir, none}`

`{read_concurrency, true | false}`

`{read_concurrency, true | false}`

ets - przegląd funkcji

`ets:new(Name, [Option]).`

`Type = ordered_set | set | bag | duplicate_bag`

`Access = private | protected | public`

`named_table`

`{keypos, Position}` albo `{keypos, #RecordName.FieldName}`

`{heir, Pid, Data} | {heir, none}`

`{read_concurrency, true | false}`

`{read_concurrency, true | false}`

`compressed`

ets - przegląd funkcji

`ets:delete(Table).`

ets - przegląd funkcji

`ets:delete(Table).`

`ets:insert(Table, Data).`

ets - przegląd funkcji

ets:delete(Table).

ets:insert(Table, Data).

ets:lookup(Table, Key).

ets - przegląd funkcji

ets:delete(Table).

ets:insert(Table, Data).

ets:lookup(Table, Key).

ets:first(Table).

ets - przegląd funkcji

ets:delete(Table).

ets:insert(Table, Data).

ets:lookup(Table, Key).

ets:first(Table).

ets:next(Table, Key).

ets - przegląd funkcji

ets:delete(Table).

ets:insert(Table, Data).

ets:lookup(Table, Key).

ets:first(Table).

ets:next(Table, Key).

ets:last(Table).

ets:prev(Table, Key).

ets - przegląd funkcji

ets:delete(Table).

ets:insert(Table, Data).

ets:lookup(Table, Key).

ets:first(Table).

ets:next(Table, Key).

ets:last(Table).

ets:prev(Table, Key).

'\$end_of_table'

ets - przegląd funkcji

`ets:match(Table, Pattern).`

ets - przegląd funkcji

`ets:match(Table, Pattern).`

`ets:match_object(Table, Pattern).`

ets - przegląd funkcji

`ets:match(Table, Pattern).`

`ets:match_object(Table, Pattern).`

Przykłady:

`Pattern = {'$1', '$2', '_', '$2'}`

`Pattern = {'$123', ok, 15, '$23', '_'}`

ets - przegląd funkcji

`ets:match(Table, Pattern).`

`ets:match_object(Table, Pattern).`

Przykłady:

`Pattern = {'$1', '$2', '_', '$2'}`

`Pattern = {'$123', ok, 15, '$23', '_'}`

`ets:match_delete(Table, Pattern).`

ets - przegląd funkcji

```
[{{'$1','$2',<<1>>,'$3','$4'},  
{'andalso',{ '>', '$4', 150 }, { '<', '$4', 500 }},  
{'orelse',{ '==', '$2', meat }, { '==', '$2', dairy }}}],  
['$1']],  
{{'$1','$2',<<1>>,'$3','$4'},  
{'<', '$3', 4.0 }, {is_float, '$3'}],  
['$1']]
```

ets - przegląd funkcji

ets:fun2ms(fun({X,Y}) when X < Y -> X+Y end).

[{ {'\$1','\$2'} , [{'<','\$1','\$2'}], [{'+','\$1','\$2'}] }]

ets - przegląd funkcji

ets:fun2ms(fun({X,Y}) when X < Y -> X+Y end).

[{ {'\$1','\$2'} , [{'<','\$1','\$2'}], [{'+','\$1','\$2'}] }]

[Clause]

Clause = {InitialPattern, Guard, ReturnedValue}

ets - przegląd funkcji

ets:fun2ms(fun({X,Y}) when X < Y -> X+Y end).

[{ {'\$1','\$2'} , [{'<','\$1','\$2'}], [{'+','\$1','\$2'}] }]

[Clause]

Clause = {InitialPattern, Guard, ReturnedValue}

ograniczenia: własne funkcje, wiele argumentów

ets - przegląd funkcji

`ets:select(Table, MatchSpecification).`

ets - przegląd funkcji

`ets:select(Table, MatchSpecification).`

`ets:select_reverse(Table, MatchSpecification).`

ets - przegląd funkcji

`ets:select(Table, MatchSpecification).`

`ets:select_reverse(Table, MatchSpecification).`

`ets:select_delete(Table, MatchSpecification).`

Dets

- API praktycznie takie samo, jak ETS
- Operacje na plikach zapisanych na dysku (limit 2GB)
- Drobne różnice w stosunku do ETS w funkcjonalności

Ets vs Dets

- brak kolekcji *ordered_set*
- *brak wsparcia dla współbieżności*
- *DETS jest znaaaacalnie wolniejsze*



dets - krótki przegląd funkcji

`open_file(Name, Args)`

`{type, set | bag | duplicate_bag}`

`{file: file:name}`

`{access, read | read_write}`

`{auto_save, infinity | milliseconds}`

dets - krótki przegląd funkcji

`open_file(Name, Args)`

`{type, set | bag | duplicate_bag}`

`{file: file:name}`

`{access, read | read_write}`

`{auto_save, infinity | milliseconds}`

`close_file(Name)`

dets - krótki przegląd funkcji

`from_ets(Name, EtsTab)`

`to_ets(Name, EtsTab) -> EtsTab`

Dets dla wymagajacych - Mnesia

- Mnesia - DataBase Management System
- korzysta z DETS
- rozbudowana dystrybucja, transakcje, zapytania
- fragmentacja tablic pozwala zwalczyć ograniczenie 2gb na plik w DETS

Test



Dziekujemy za uwage

Szymon Łabuz
Tomasz Legutko