CDC to the Max!

CDC driving new technology
Who am I?

Bronco Oostermeyer

OpenEdge since 1996, everything around Progress and more
NativeScript/Angular/Node.js/TypeScript
Progress, UNIT4, WALVIS
Who are we?

Flusso

- Biggest OpenEdge consultancy firm in NL
- Progress technologies
- Open Source (Java & ServiceMix, etc),
- Web Apps (Angular2, various app platforms)
- OutSystems
Agenda

• CDC
• Use cases
• High speed search engine
(CDC) To the Max!

Since we didn’t qualify for the World Cup 😊
The Challenge (for me)

• A Use Case for CDC, new in 11.7
  • “nice feature, but what to do with it?”
• Come up with scenario which doesn’t suit a RDBMS
• Challenge you to think differently about your application
Change Data Capture

• It captures changes to data
  • ALL changes
  • Including originated from SQL

• Transparent
  • Your current tables stay exact the way they are
  • CDC data in new tables, same database
  • Configurable rather than need for programming
  • NOT trigger based
Licensing

- Standard in Enterprise DB advanced
- Add-on for
  - Enterprise DB
  - Workgroup DB
Steps

• Make dedicated storage area’s for CDC data & CDC index
• Enable the CDC feature
• Assign storage area’s
• Create policies
Add area’s

d "cdc_data":98,32;64 .
d "cdc_index":99,32;8 .

prostruct add <dbname> <st file>
proutil <dbname> -C enablecdc
area <cdc_data_area>
indexarea <cdc_index_area>
Policies

Storage Management
Manage database physical and logical storage

- Change Data Capture Tables List
- Create Change Data Capture policy

Create Change Data Capture policy

Policy name: customer_policy
Description:
Database connection: sports2000
Table: PUB.Customer
Level:
State:
- Active
- Inactive
tracking changes

After enabling CDC you get:

- _cdc-field-policy
- _cdc-table-policy

```
_Cdc-Change-Tracking
Columns [16]
  _Policy-Id [character(22)]
  _Tran-Id [integer >,>>,9]
  _Change-Sequence [int64]
  _Operation [integer >,>>,9]
  _Change-FieldMap [new x(256)]
  _Recid [recid >,>>>9]
  _Source-Table-Number [integer >,>>,9]
  _Partition-Id [integer >,>>,9]
  _Tenant-id [integer >,>>,9]
  _Version [integer >,>>,3]
  _User-Misc [character(8)]
  _User-Name [character(40)]
  _Misc [character(32)]
```
tracking changes– level 0

- 4 levels
- Just registers update event in _Cdc-Change-Tracking table:

**WARNING:** Level 0 cannot be upgraded to higher level.
tracking changes – level 1, 2 & 3
what will be tracked

(and “name” as well)
levels explained

• 1 – customer 1 has changed (w/ fieldlist)
• 2 – customer 1 has changed, these are the new values
• 3 – customer 1 has changed, these are the old & new values
cdc_customer (level 3)

```java
find first customer.
creditlimit = creditlimit + 3300.
```

<table>
<thead>
<tr>
<th>_Tran-id</th>
<th>_Time-Stamp</th>
<th>_Change-Sequence</th>
<th>_-</th>
<th>_-</th>
<th>_-</th>
<th>_Operation</th>
<th>CustNum</th>
<th>Name</th>
<th>CreditLimit</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>28298</td>
<td>2017-09-05 12:08:30:543 + 02:00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>&lt;NULL&gt;</td>
<td>66700.00</td>
<td>&lt;NULL&gt;</td>
</tr>
<tr>
<td>28298</td>
<td>2017-09-05 12:08:30:543 + 02:00</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>&lt;NULL&gt;</td>
<td>70000.00</td>
<td>&lt;NULL&gt;</td>
</tr>
</tbody>
</table>
CDC Summary

• Capture all changes transparently
• dedicated table for storing changes
• a _change-sequence_ field to keep track off
CDC Use cases

- Friendly alternative for auditing (sort of)
- Off-loading changes to satellite systems
  - Search engine
  - Back-end for web
  - Data warehouses
  - High Availability/-Scalability
  - ...

ITkracht_van_Flusso
But why?

• Assume an *System of Record* (f.e. an ERP)
• You know all the users?
• Do you still do when you want to go to the Web?
• Do those complicated searches really perform?
• Can you calculate the requirements for new hardware?
What if?

- You can scale horizontally (dynamically)
- In the cloud
- You could deliver real 24/7 for web users
- You can prevent your back office being locked out by web traffic
- And still use your ERP as is?
Put it to use

- OpenEdge database
  - 2 tables: person & address
  - 14,412,063 different person records (true names)
  - 256,221 unique last names
  - 1 address record per person
  - divided over 388 municipalities (cities)
  - based on true data on NL
Schema

For sake of this presentation:
a very simple schema

person
  id: int64
  lastname: char

address
  id: int64
  person_id: int64
  city: char
### db contents

#### Select distinct last names, count(*) from pub.person where last

<table>
<thead>
<tr>
<th>Lastname</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jonse van Vuuren</td>
<td>1</td>
</tr>
<tr>
<td>Jonse van Vuuren</td>
<td>1</td>
</tr>
<tr>
<td>Janse Vreeling</td>
<td>1</td>
</tr>
<tr>
<td>Janselijn</td>
<td>5</td>
</tr>
<tr>
<td>Jonsemo</td>
<td>336</td>
</tr>
<tr>
<td>Jansen</td>
<td>73533</td>
</tr>
<tr>
<td>Jansen Adriaans</td>
<td>1</td>
</tr>
<tr>
<td>Jansen Duighuizen</td>
<td>28</td>
</tr>
<tr>
<td>Jansen Eijkens Slueters(y)</td>
<td>6</td>
</tr>
<tr>
<td>Jansen Eupe</td>
<td>42</td>
</tr>
<tr>
<td>Jansen Fraiksen</td>
<td>5</td>
</tr>
<tr>
<td>Jansen Heijmoijer(y)</td>
<td>12</td>
</tr>
<tr>
<td>Jansen van Heering</td>
<td>6</td>
</tr>
</tbody>
</table>

#### Select distinct city, count(*) from pub.address group by city

<table>
<thead>
<tr>
<th>City</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphen aan den Rijn</td>
<td>91894</td>
</tr>
<tr>
<td>Alphen-Chaam</td>
<td>8486</td>
</tr>
<tr>
<td>Ameland</td>
<td>3065</td>
</tr>
<tr>
<td>Amersfoort</td>
<td>130218</td>
</tr>
<tr>
<td>Amstelveen</td>
<td>75339</td>
</tr>
<tr>
<td>Amsterdam</td>
<td>712902</td>
</tr>
<tr>
<td>Apeldoorn</td>
<td>135935</td>
</tr>
<tr>
<td>Arnhem</td>
<td>131367</td>
</tr>
<tr>
<td>Assen</td>
<td>56994</td>
</tr>
<tr>
<td>Asten</td>
<td>14102</td>
</tr>
<tr>
<td>Baarle-Nassau</td>
<td>5617</td>
</tr>
<tr>
<td>Breda</td>
<td>26295</td>
</tr>
</tbody>
</table>
the query

```sql
for each address where address.city = "Amsterdam" no-lock,
    first person where person.id = address.person_id
        and person.lastname = "Jansen" no-lock:
    i = i + 1.
end.
```

approx.: (after clean db start ~1min)
RDBMS vs NoSQL

• joined queries not fast (RDBMS)
  • stuck to db structure

• NoSQL databases are very fast
  • but lack transactional capabilities (mostly)
  • Joins are not always supported

• MongoDB, Redis, Cassandra, ElasticSearch
  • and CouchDB, Hbase, Accumulo, and ... and ...
  • https://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis
Enter MongoDB

- `Progress OpenEdge`
  - `person`
  - `cdc_person`
  - `address`
  - `cdc_address`

- `mongoDB`

```json
[{  "id" : 6943896,    "lastname" : "Jansen",    "address" : {      "city" : "Amsterdam"    }  }, ...]
```
How keep you MongoDB up-to-date?

```json
{
  "changeseqid": 136,
  "info": {
    "id": 32944039,
    "lastname": "test_qwerty",
    "address": {
      "city": "Aalsmeer"
    }
  }
}
```
cdc_person

<table>
<thead>
<tr>
<th>Columns</th>
<th>Indexes</th>
<th>Primary Key</th>
<th>Foreign Key</th>
<th>Preview</th>
<th>Row Count</th>
<th>Privileges</th>
<th>Partition Key</th>
<th>OpenEdge Table</th>
<th>OpenEdge Columns</th>
<th>OpenEdge Indexes</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Tran-id</td>
<td>_Time-Stamp</td>
<td>_Change-Sequence</td>
<td>_Continuation-Position</td>
<td>_ArrayIndex</td>
<td>_Fragment</td>
<td>_Operation</td>
<td>id</td>
<td>lastname</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469310</td>
<td>2017-07-19 08:26:09:154 + 02:00</td>
<td>132</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32944039</td>
<td>test_jansen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469376</td>
<td>2017-07-19 12:25:40:134 + 02:00</td>
<td>133</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>32944039</td>
<td>test_qwerty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469423</td>
<td>2017-07-19 12:30:50:992 + 02:00</td>
<td>134</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>32944039</td>
<td>test_qwerty1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469424</td>
<td>2017-07-19 12:33:12:166 + 02:00</td>
<td>135</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>32944039</td>
<td>test_qwerty2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469425</td>
<td>2017-07-19 13:07:46:241 + 02:00</td>
<td>136</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>32944039</td>
<td>test_qwerty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469432</td>
<td>2017-07-19 14:08:54:157 + 02:00</td>
<td>137</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>32944039</td>
<td>test_123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38469433</td>
<td>2017-07-19 14:12:01:748 + 02:00</td>
<td>138</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32944041</td>
<td>test_jansen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What more?

• The target system needs to keep track of processed changes
• The target system needs to be able to report last change id
• a search api
Technology Stack

• OpenEdge Change Data Capture
• Node.js (TypeScript)
  • express
  • mongodb (connector)
• MongoDB
• Angular (TypeScript)
  • Bootstrap
Landscape

dispatcher

expressjs

mongoDB

cdc_person

cdc_address

https://github.com/bfbliz

expressjs

https://github.com/bfbliz

Angular

ITkracht_van_Flusso
Demo
The code

- https://github.com/bfv/pc2017
- OpenEdge dispatcher (/cdc)
  - httpclient
- Node JS (/search)
  - Express, MongoDB
- Angular (/webui)
  - bootstrap
Node /search code

GET http://localhost:4220/search?lastname=Jansen&city=Amsterdam

server.ts

```typescript
this.dbconnector = new MongoConnector();

private routes() {
    const routes: Routes = new Routes(this.dbconnector);
    this.router.get('/', routes.home);
    this.router.get('/search', routes.search);
    this.router.get('/changeSeqId', routes.changeSeqId);
}
```

router.ts

```typescript
export class Routes {

    private static connector: DbConnector;

    constructor(connector: DbConnector) {
        Routes.connector = connector;
    }

    public search(req: Request, res: Response, next: NextFunction) {
        let persons: Person[];
        console.log('search');
        let lastname = req.param('lastname');
        let city = req.param('city');

        Routes.connector.searchPersons(lastname, city, (data) => {
            persons = <Person[]> data;
            res.contentType('application/json');
            res.send({ route: '/search', persons: persons, status: 'ok' });
        });
    }
}
```
connector code

dbconnector.ts

```typescript
import { Person } from './types/person';

export interface DbConnector {
  init(): Promise<boolean>;
  searchPersons(lastname: string, city: string, callback): void;
  upsert(person: Person): void;
  updateChangeSeqId(table: string, changeSeqId: number, callback): void;
  bulkCreate(persons: Person[], callback): void;
  getCities(callback): void;
  getChangeSeqId(callback): void;
}

{
  "lastname": "Jansen",
  "address.city": "Amsterdam"
}
```
mongoconnector.ts

```typescript
export class MongoConnector implements DbConnector {
  public searchPersons(lastname: string, city: string, callback): void {
    let persons: Person[] = [];
    let criteria = {};

    if (lastname && lastname != '') {
      criteria["lastname"] = lastname;
    }

    if (city && city != '') {
      criteria["address.city"] = city;
    }

    console.log('begin search:', criteria);

    this.persons.find(criteria).toArray().then(data => {
      console.log('done');
      persons = <Person[]> data;
      callback(persons);
    });
  }
}
```
Result

```json
[{
    "id" : 6943896,
    "lastname" : "Jansen",
    "address" : {
        "city" : "Amsterdam"
    }
}, ...]
```
But...

- cdc data is in normal tables (delta.df might wipe data)
- cdc configuration is dumped/loaded separately
Conclusion

• CDC is pretty easy to configure
• Resulting cdc table are readable
• Use change_sequence to keep track
• The possibilities are limited by your own fantasy
• Highly scalable cloud systems can be fed by CDC
• Enables new technology w/o starting all over
• Last advise: *Think big, start small*
Questions?
OpenEdge on Slack!

https://tinyurl.com/slackOpenEdge
follow us on: