

**They call us crazy,
but we store Contacts in Tracker**

Mathias Hasselmann, Openismus



What is this about?

- **QtContacts** – easy to use, cross-platform address book API, Nokia uses it on Symbian and Harmattan
- **Tracker** – GNOME's version of a RDF tuple store

“Semantic Desktop”



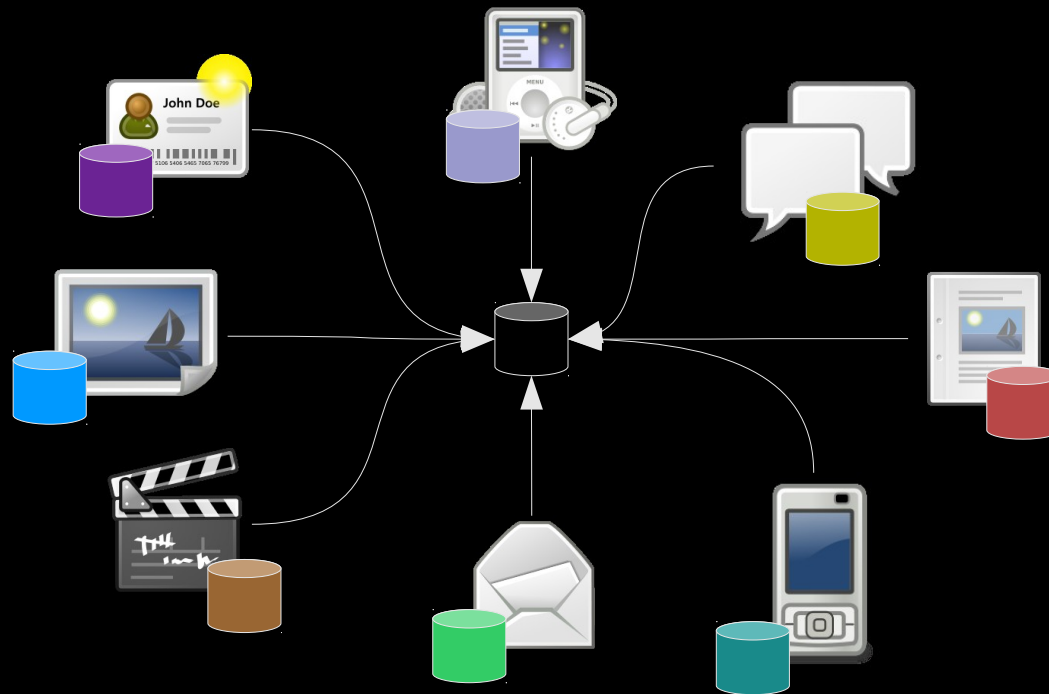
Traditional Desktop

Lots of interesting and useful information
spread over many detached data sources.
Not accessible.



Miners and Harvesters

Aggregation: Many independent data stores.
Harvesters monitor them and update an additional database.



Miners and Harvesters

Positive

- no changes to existing applications

Negative

- waste of CPU cycles, I/O cycles, and memory
 - code duplication, unreliable miners
 - latency and other synchronization issues
- only few applications actually use the collected data

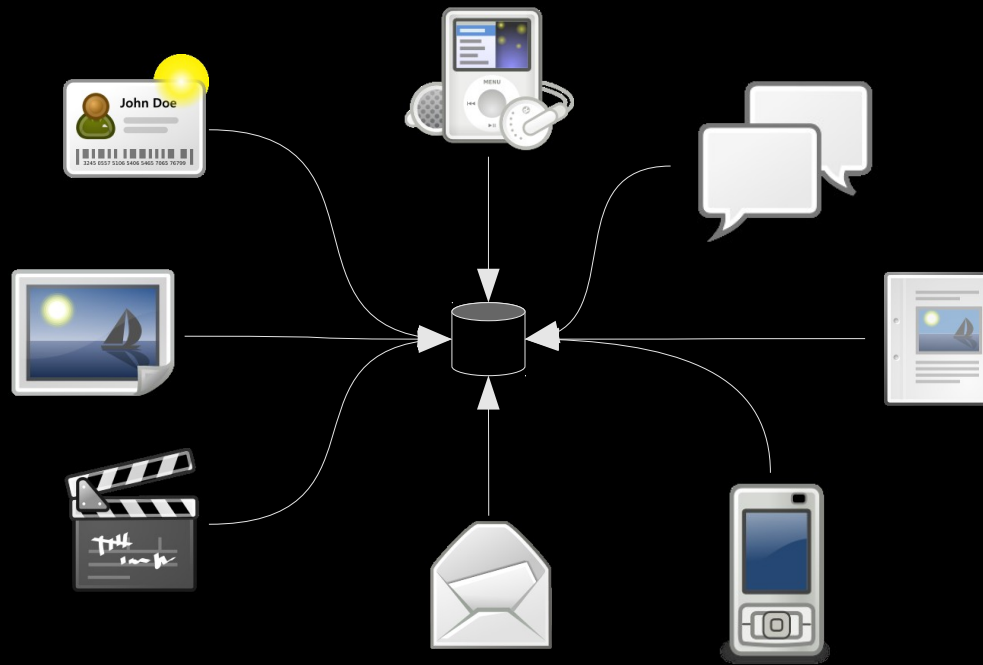
Perception

“Beagle, Tracker, Zeitgeist, ... are useless bloat”



Semantic Desktop

How about applications putting (relevant) information into **one semantic data store**?



Semantic Desktop

Negative

- applications must be changed

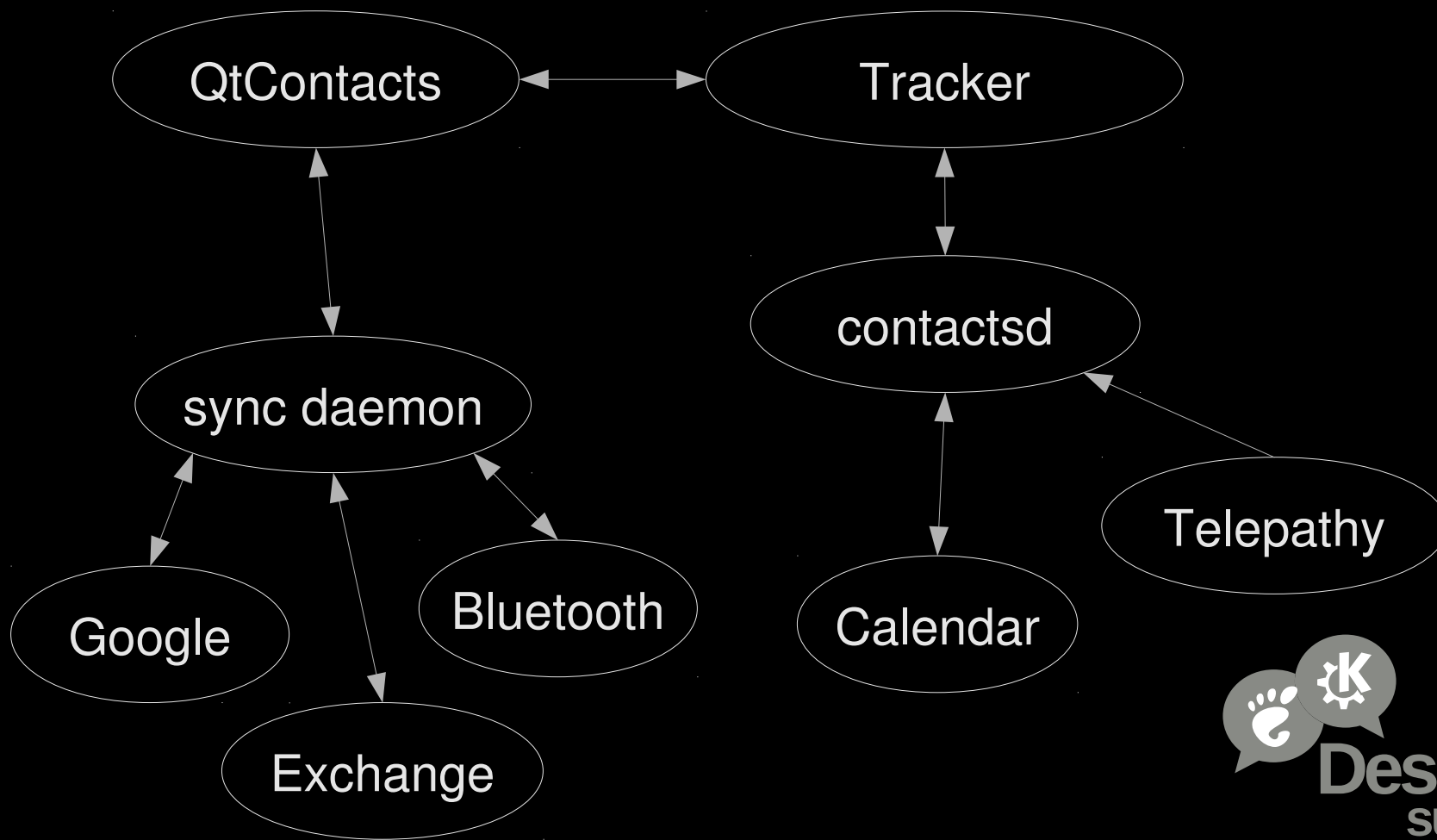
Positive

- lower resource usage
- less code duplication, perfect meta data
- minimal latency, no synchronization issues
- perfectly integrated applications

Perception

“This is awesome” - hopefully





RDF, Sparql

- Well defined, interoperable standards.
- It is science! Very smart people research it!
- Countless papers about properties, limitations, algorithms.

vs.

random, ad-hoc, NIH solution



RDF Data Model

subject predicate object .

e.g. <nco:default-contact-me> a nco:PersonContact

- resources identified by IRI
- classes organized in ontologies
- predicates and ranges defined by classes
- multi inheritance



RDF Contacts

- NEPOMUK ontology (with a few “bug fixes”)

```
<urn:uuid:1234...> a nco:PersonContact ;  
  nco:nameGiven "Hans" ;  
  nco:nameFamily "Zwergl" ;  
  nco:hasAffiliation <urn:uuid:50da...> ;  
  nco:websiteUrl <http://zwer.gl/> .  
  
<urn:uuid:50da...> a nco:Affiliation ; rdfs:label "Home" ;  
  nco:hasPhoneNumber <urn:x-maemo-phone:...> .  
  
<urn:x-maemo-phone:...> a nco:CellPhoneNumber ;  
  nco:phoneNumber "+49-172-55443322" ;  
  maemo:localPhoneNumber "55443322" .
```



Sparql Queries

- SPARQL Algebra – quite similar to relational algebra
 - projections, restrictions, filters

SELECT

```
?contact nco:phoneNumber (?tel)
```

WHERE {

```
?contact a nco:PersonContact .
```

```
?contact nco:hasAffiliation [ nco:hasPhoneNumber ?tel ] .
```

```
FILTER (fn:ends-with (maemo:localPhoneNumber (?tel), "334455")) .
```

```
}
```



Sparql Updates

- INSERT and DELETE, no update statement
(well, tracker has INSERT OR REPLACE)

```
DELETE {  
    ?contact nco:hasAffiliation ?affiliation  
} WHERE {  
    ?affiliation rdfs:label "Work"  
}
```

```
INSERT {  
    _:contact a nco:PersonContact ;  
    nco:birthDate "1990-01-01"^^xsd:date ;  
    nco:fullname "Example Contact" .  
}
```



QtContacts API

- make the common use cases trivial, no point in learning SPARQL for them
- based on careful evaluation of libebook
- asynchronous and synchronous API, notifications
 - contact manager and action plugins
 - contacts organized as collection of details
 - details described by POD classes and schema
- detail linking to mark (e.g origin of presence or avatars)
 - trivial to add new details and detail actions
 - contact filters, fetch hints
 - partial contact saving
 - contact relationships



Presence

- nco:hasIMAddress, nco:imPresence, nco:imCapability, ...
- contactsd plugin mirrors presence status from Telepathy to tracker

Advantages

- we can have queries on presence status
- no additional step to apply presence status to contacts
 - applications only wake up from contact changes, not on each Telepathy change

Problems

- with direct tracker access we lost transient property support, presence data is written do disk – very bad!



Merging, Unmerging

```
INSERT {
  _:contact a nco:PersonContact .

  GRAPH <first-origin> {
    _:contact nco:hasEmailAddress <...> .
    _:contact nco:hasPostalAddress <...> .
  }

  GRAPH <second-origin> {
    _:contact nco:hasIMAddress <...> .
  }
}

SELECT ?g ?p ?v {
  GRAPH ?g { <contact> ?p ?v }
}
```



Phone number IRIs

Wanted

- content based IRIs for fast lookup, to avoid duplications

Problem

- on sync different variants of same contact with varying quality
- can't just store the "best" variant, since the origin might not support all details and such → sync, resync problems
- a data store shall store what you throw at it and not be too smart

`urn:x-maemo-phone:voice,cell:+49-172-55443322`



Scalar Selects

```
SELECT ... WHERE { ... OPTIONAL {  
    ?contact nco:hasPostalAddress ...  
} }
```

→ left join in sqlite, horrible performance

```
SELECT ?contact  
    (SELECT fn:concat (nco:streetAddress (?a), '\x1f',  
                    nco:postalCode (?a), ...)  
    ?contact nco:hasPostalAddress ...)  
WHERE {  
}
```

→ scalar select in sqlite, awesome performance



Garbage Collection

- when updating or deleting contacts resource links get removed for performance, mainly nco:hasAffiliation
- leaves abandoned resources, wastes disk space, pollutes indexes, degrades performance

garbage collection plugin in contactsd:

- register a named GC query and increase its weight with each update
- upon weight threshold or timeout (often expensive) GC query is run



Links

<https://gitorious.org/qtcontacts-tracker>

<https://gitorious.org/cubi>

<https://maemo.gitorious.org/maemo-af/qsparql>

<http://doc.qt.nokia.com/qtmobility-1.2/contacts.html>

<http://www.w3.org/RDF/>

<http://www.w3.org/TR/sparql11-query/>

<http://www.w3.org/TR/sparql11-update/>

<http://developer.gnome.org/ontology/unstable/>

