A Benchmark for Context Data Management in Mobile Context-Aware Applications*

Nadine Fröhlich, Steven Rose

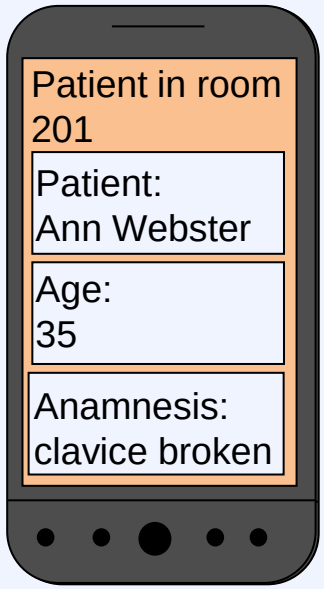
Thorsten Möller, Heiko Schuldt

University of Basel, Switzerland

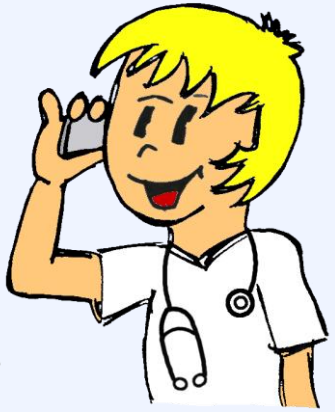
13th of September 2010

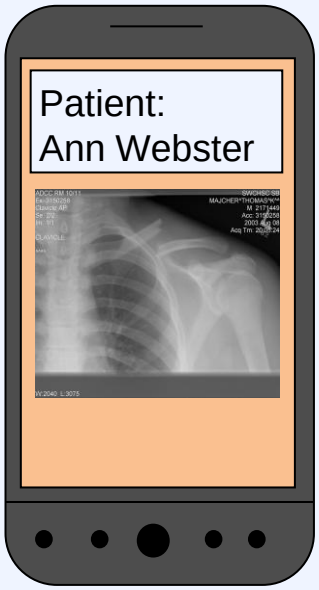
*This work is partially funded by the Hasler Foundation (project LoCa)

Motivation: Scenario (Patient visit)



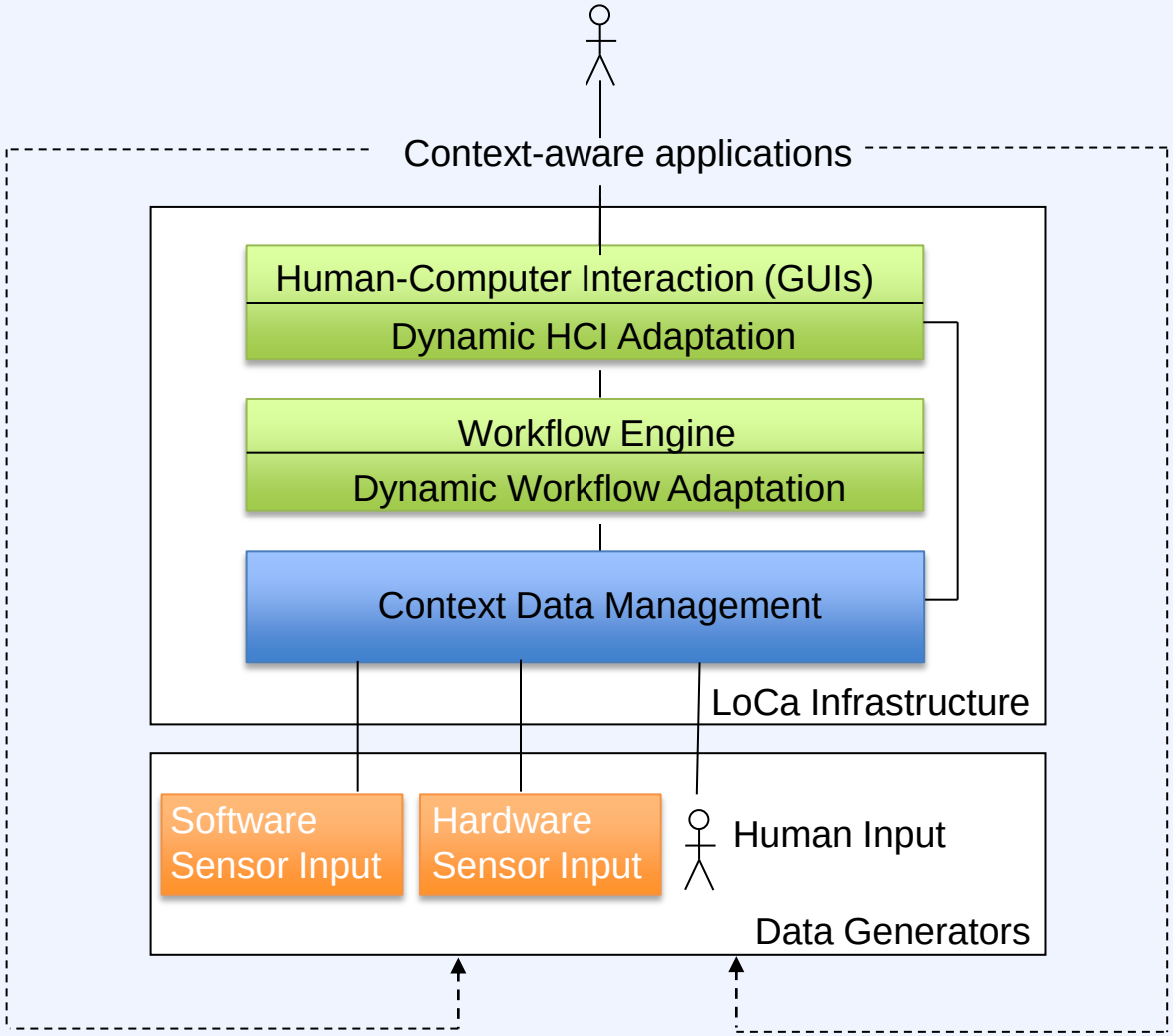
Entering Room 201





automated context aware adaptation of application workflow and GUI
→ ease application handling, and avoid errors

LoCa Architecture



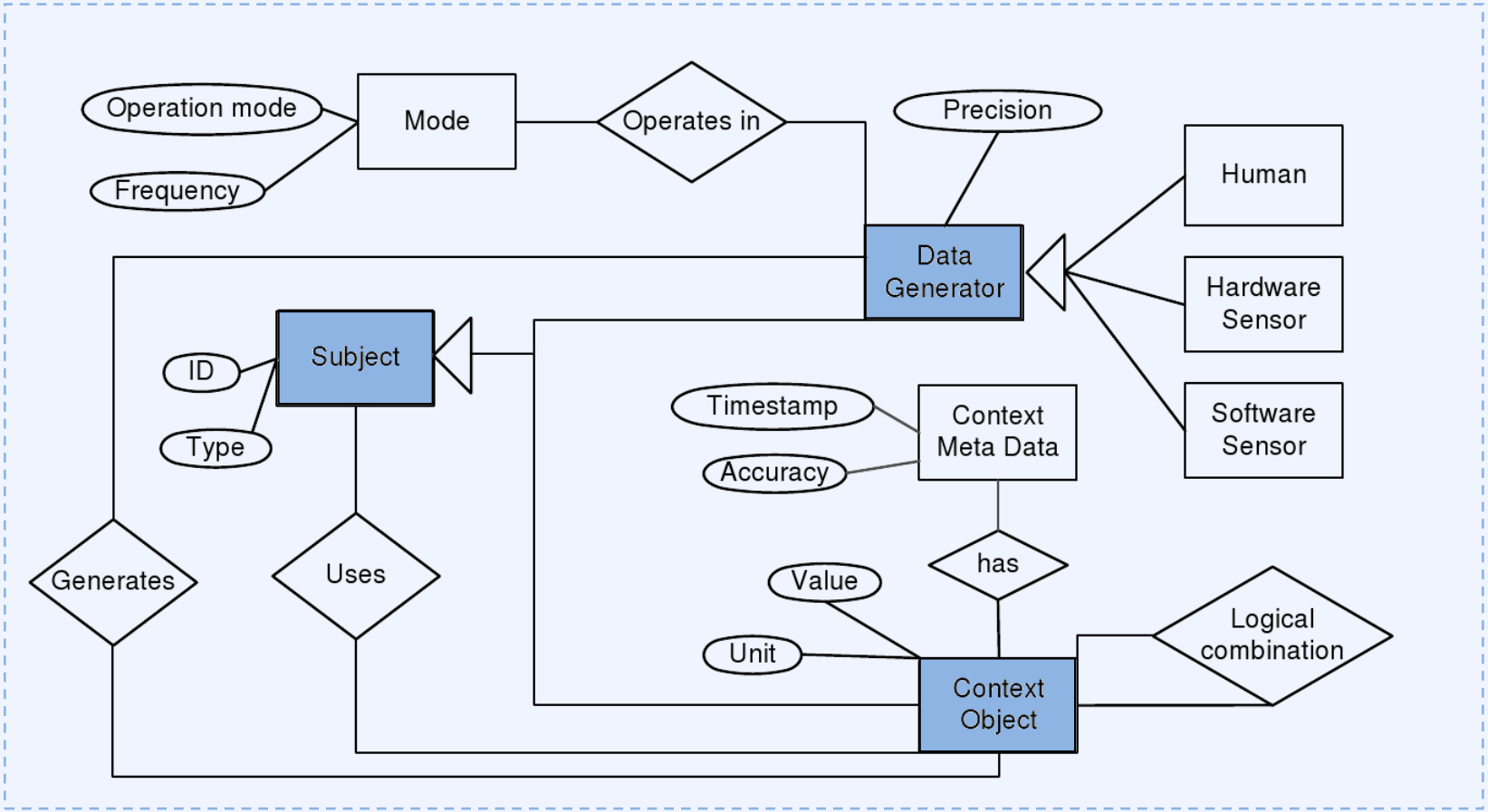
No convenient benchmarks for mobile phones available → own benchmark

Benchmark Setup

- RDBMS: SQLite, H2, MySql
- OODB: db4o
- Triple store: Sesame – OWLIM

Data model

- Context is any information that can be used to characterize the situation of a subject. A subject is a person, place, or object that is considered relevant to the interaction between a user and an application […]. Day99



Estimating data load

Entity	#entities for medium sized hospital year	#entities for one person within a year
DataGenerator	30.500	851
Human	5.000	1
HardwareSensor	25.000	833
SoftwareSensor	500	17
Mode	5.000	851
ContextObject	925.000.000	185.000
ContextMetaData	925.000.000	185.000
LogicalCombination	925.000	185
Subject (patients)	27.000	1.800

time dependent

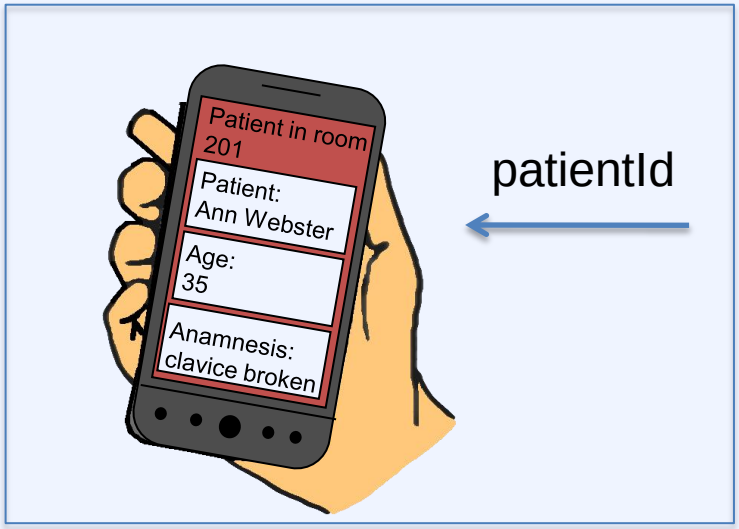
- #entities for one person within a quarter (year/4)
- #entities for one person within a week (year/52)

Benchmark queries (I)

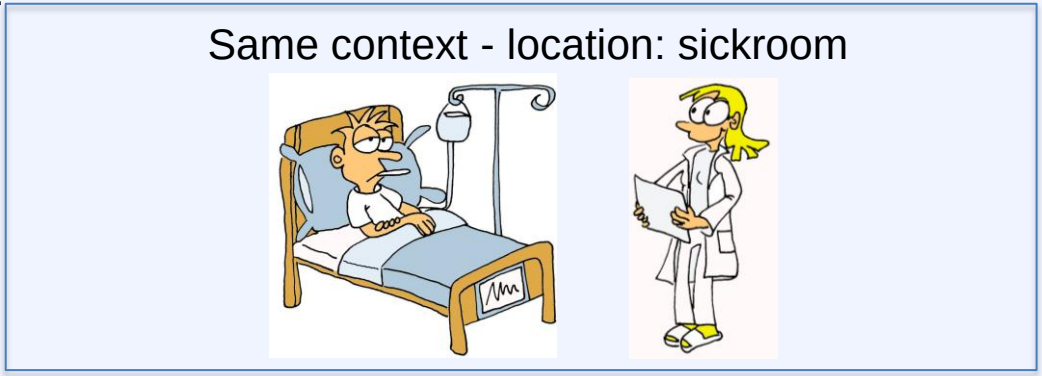
1. Return a subject by a given id
2. Return last recorded context object of a given type for a subject s
3. Return context object of a given for a subject in a given time interval (day, week, year)
4. Find last recorded context object of type x belonging to subject s generated by generator g
5. Find all data generators including type and precision
6. Find subject of type x sharing a context object with a given subject

Benchmark queries (I)

1. Return a subject by a given id



6. Find subject of type x sharing a context object with a given subject

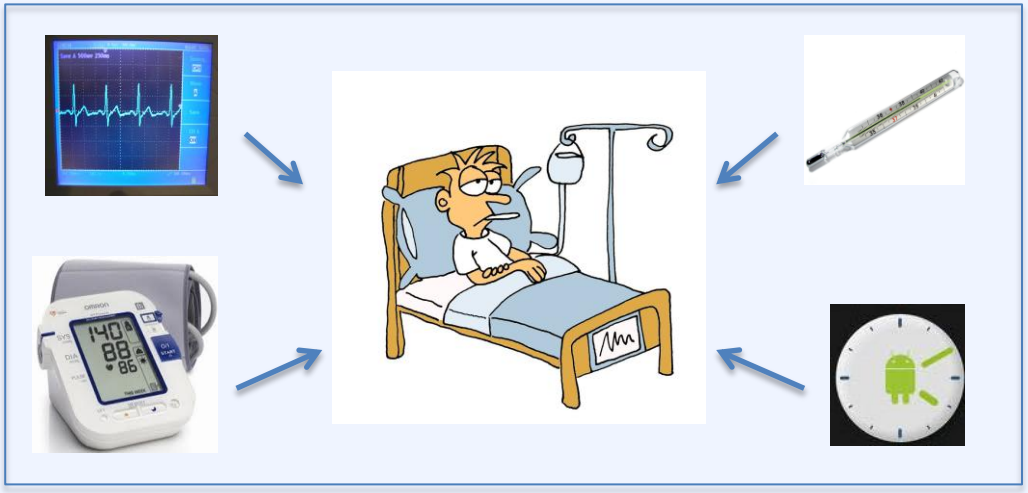


Benchmark queries (II)

7. Find all available types of context object available for a given subject in alphabetic order
8. Find all logical combinations of context data for a given subject
9. Find the number of data generators per subject
10. Insert a context object
11. Update all context objects belonging to one logical combination
12. Delete a context object

Benchmark queries (II)

9. Find the number of data generators per subject



Performance Metrics

- Metrics for single queries
  - Average Query Execution time (10 runs) (metered)
  - Min/max query execution time (metered)
  - Queries per second (computed)

- Metrics for query mixes
  - Average Execution time (3 runs)

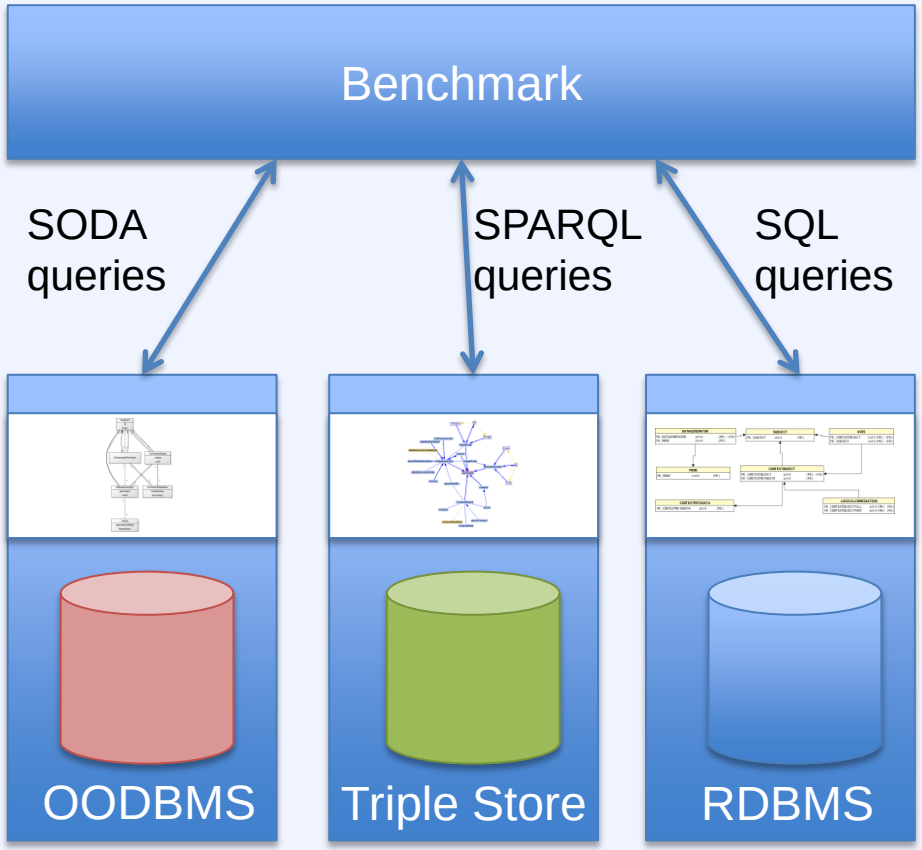
Queries distribution in query mixes

- updates and deletes rarely used
- frequent inserts of sensor data (concurrently to selects)
- query for subject by id (Q1) and connections between subjects (Q6) most often needed selection

Query	Percentage
Query1	10 %
Query2	7 %
Query3	7 %
Query4	6 %
Query5	3 %
Query6	13 %

Query	Percentage
Query7	7 %
Query8	6 %
Query9	4 %
Query10	36 %
Query11	0.5%
Query12	0.5%

Benchmark implementation



Implementation of queries per data store (same parameters for queries)

Implementing the data model per data store

Generation of test load per data store (same data basis for all data stores)

Benchmark results

- Query mixes – embedded data stores

System	Week	Quarter	Year
H2 emb	26s	1300s	23251s
SQLite emb	386s	x	x
db4o emb	511s	21100s	x

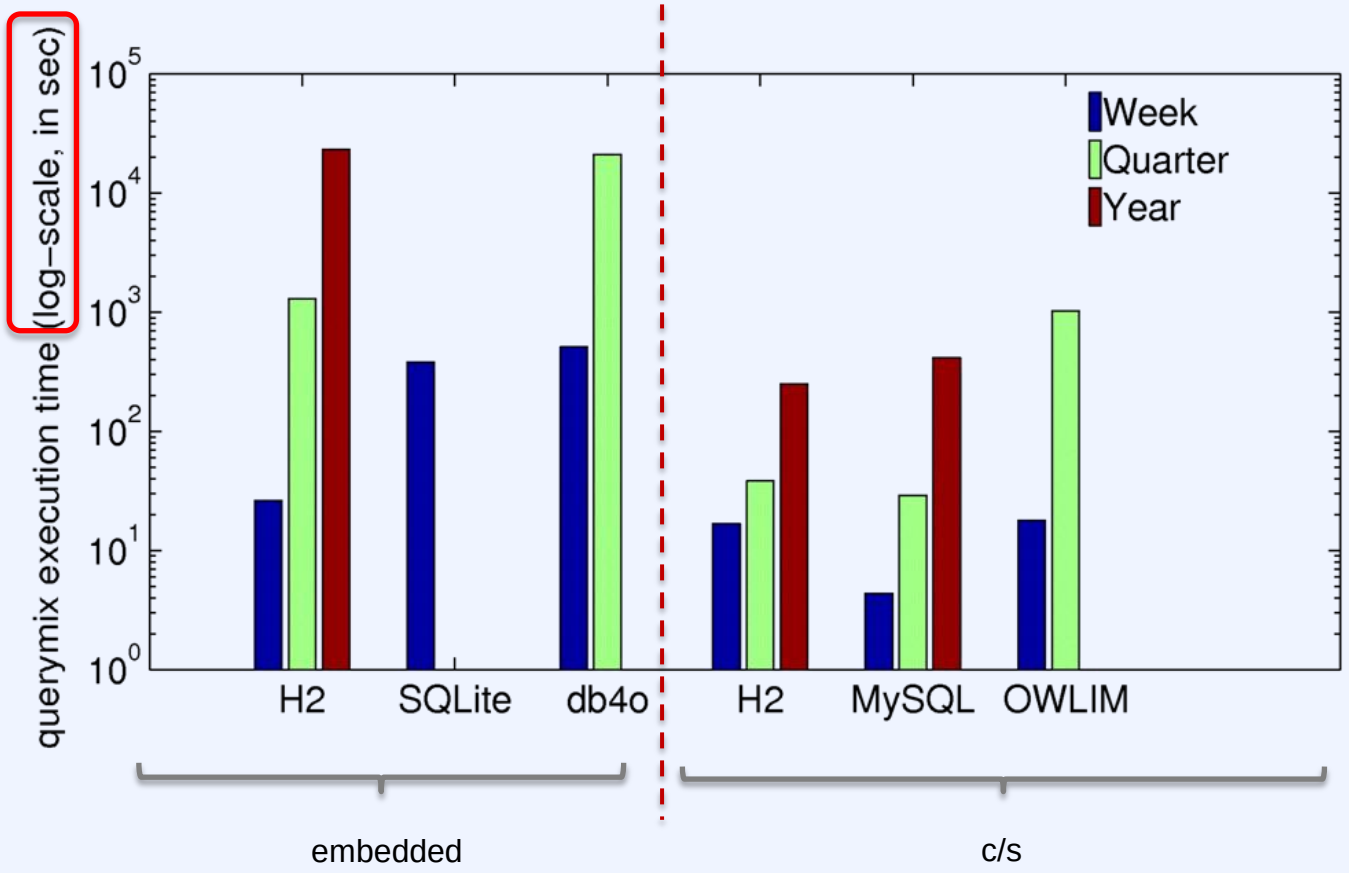
6,5 h

- Query mixes – data stores in c/s mode

System	Week	Quarter	Year
H2 c/s	17s	38s	251s
MySQL c/s	4s	29s	418s
OWLIM c/s	18s	1025s	x
db4o c/s	x	x	x

x – execution crashed (jvm heap, memory, …)

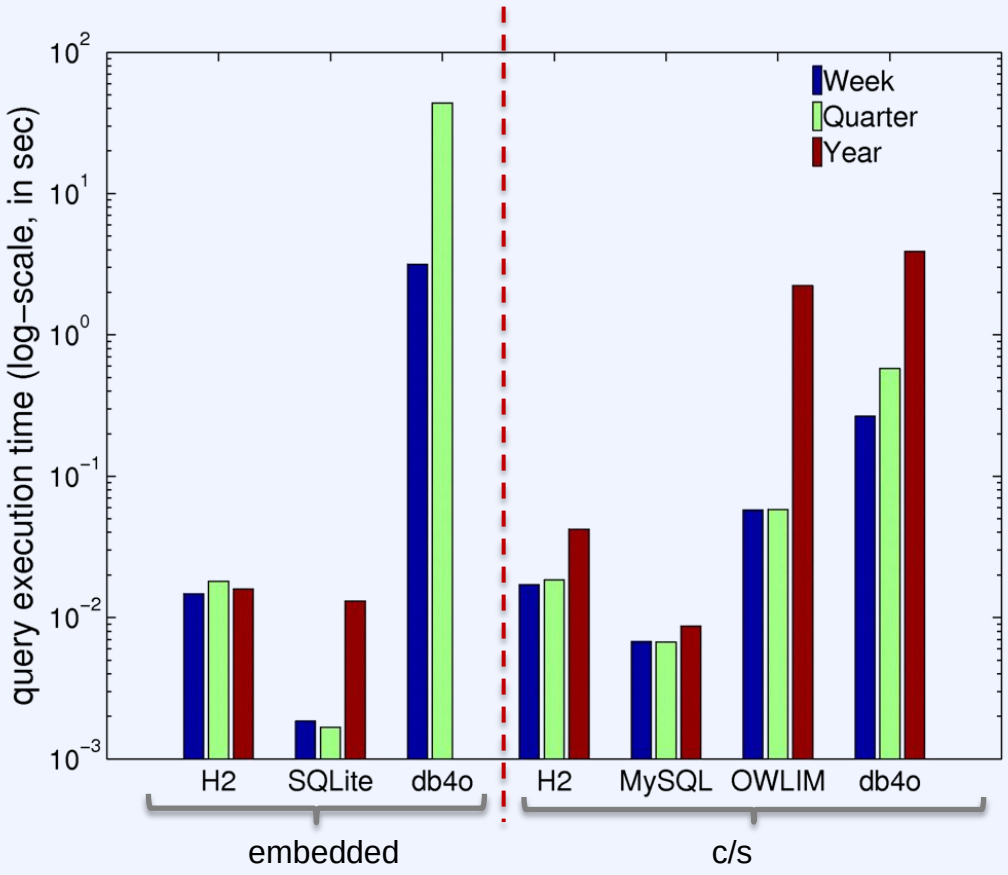
Results - query mixes



- RDBMS in c/s mode perform best
- OWLIM and db4o could not finish year run

Results - Query 1 (single query)

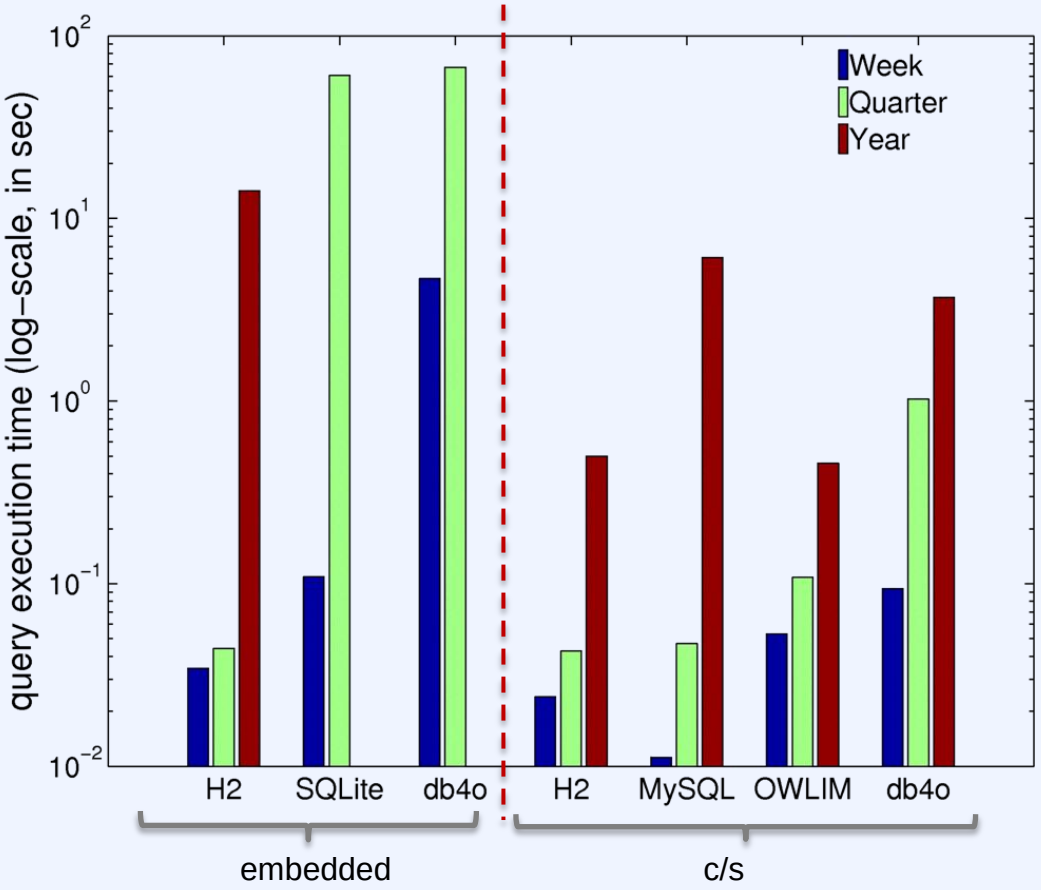
Return a subject by a given id



- SQLite performs best
- only db4o embedded could not finish year run

Results - Query 6 (single query)

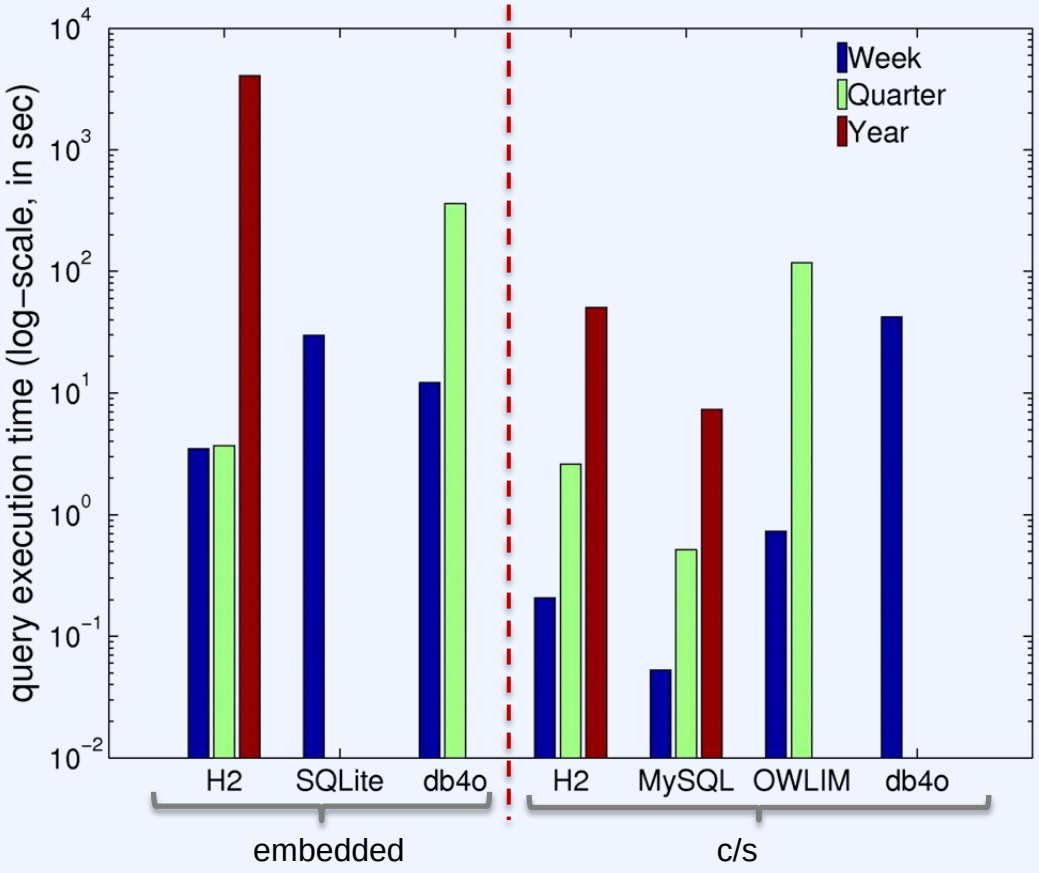
Find subject of type x sharing a context object with a given subject



- OWLIM performs best for year run
- RDBMS slow up for year run

Results - Query 9 (single query)

Find the number of data generators per subject



- complex query, MySql performed best
- complete results only for RDBMS

Summary and Outlook …

- Benchmark Goal:
  - Find a data store convenient for context management on mobile devices
  - Consider different paradigms
  - Generic comparison of data stores without preference on a special application or store

- Results for our model and the chosen setting:
  - In query mixes RDBMS on average faster than triple stores faster than OODBMS
  - H2 best results regarding average performance in query mix
  - Depending on query type perform triple stores or OODBMS better than RDBMS
  - Remote access faster than local installation on mobile phone
  - Local management of data set for a year not meaningful

… Summary and Outlook

- Limitations of mobile devices still huge compared to desktop PCs (JVM heap, memory, CPU, …)

- Different limitations of different databases
  - SQLite: limited optimization, no referential integrity by default
  - db4o: performs bad for hierarchies and collections, lack of query constructs
  - OWLIM: supports SPARQL not SPARQL update → no support of update operation

- Outlook
  - Adapt workflows and graphical user interfaces according to the context changes

Thank you

- Questions?