



# The Infinite filing Cabinet

---

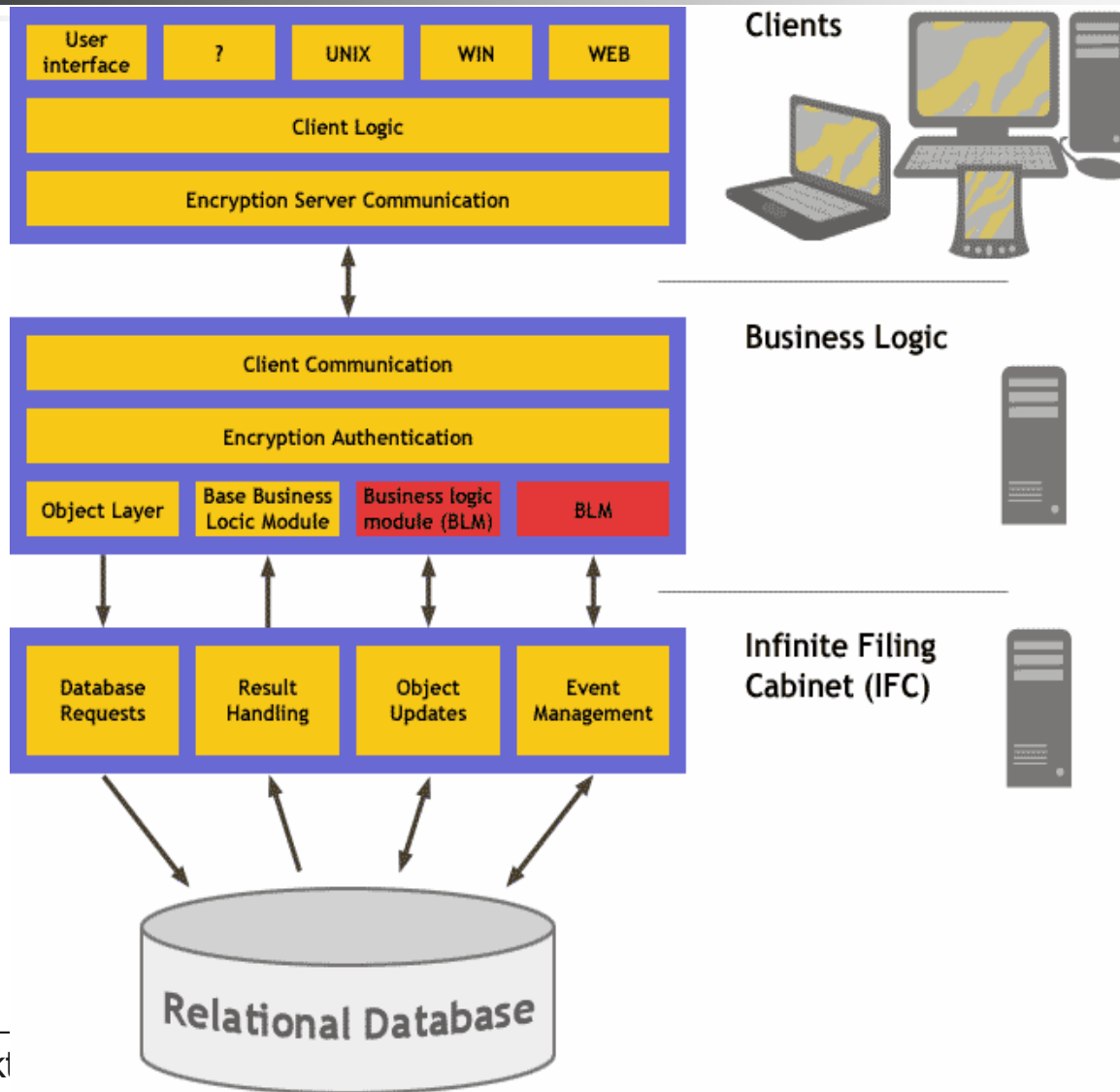
An object related database with a twist

PyCon, March 27, 2003

Jacob Hallén

[jacob@strakt.com](mailto:jacob@strakt.com)

# CAPS™ Architecture





# Features of the IFC™

---

- Object-relational translation
- Cacheing of objects
- Query object structure - SQL query translation



# Features (2)

---

- Subscriptions
  - Object
  - Query
- Events
  - Object transition
  - Time
  - Rule



# Special Properties

---

- **No delete operation**
  - Can be done through an off-line procedure
- **Audit trail**
  - Transactions are logged
  - Previous object state is saved



# Plattform

---

- PostgreSQL
- Python
- Psycopg
- Twisted Internet
- OpenSSL/pyOpenSSL

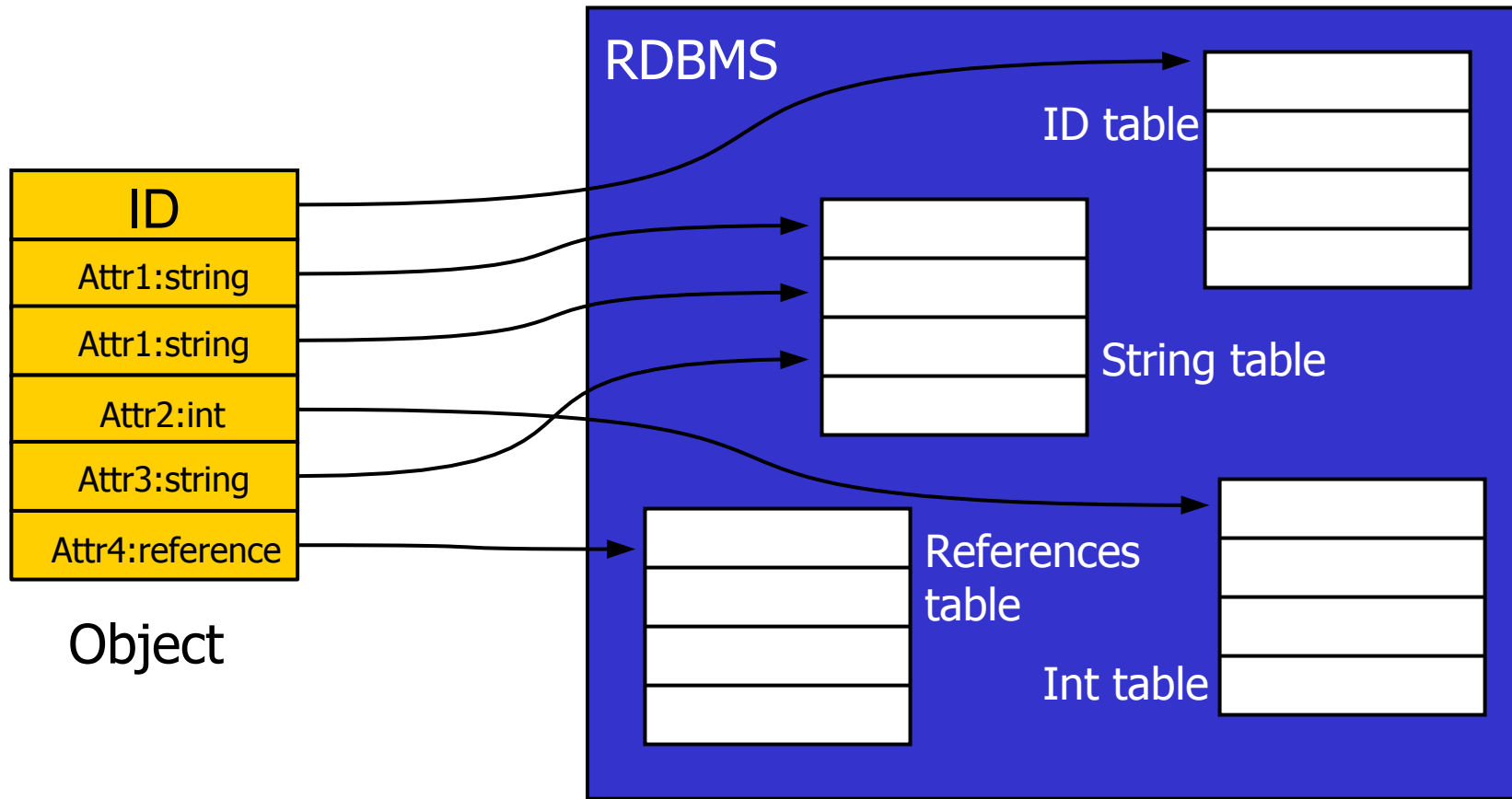


# Object <-> Relational Translation

---

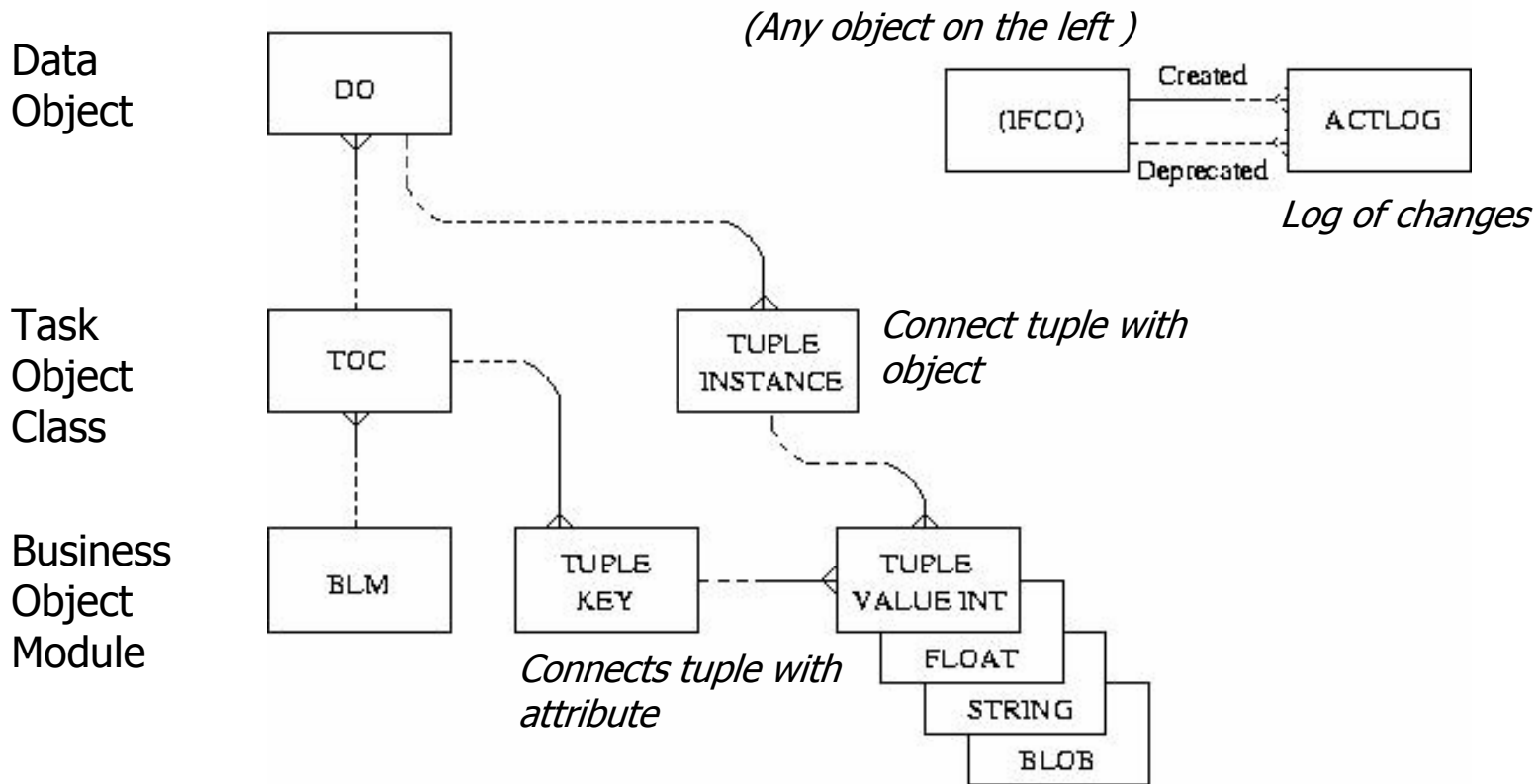
- Objects entered into the database are picked apart and stored as its smallest components
- Requested objects are assembled from components into the object
- Support tables keep track of which data goes where and what labels stick to each

# Object Decomposition



# How (De)composition Works

## IFC Entity Relations Graph





# Attributes

---

- An object concept
  - Single type sequence of values
- Types
  - String (indexable and non-indexable)
  - Boolean, Integer, Float
  - Time/date
  - BLOB (any binary data + MIME type)
  - Object Reference (type restricted)



# Cache

---

- Object construction is not a trivial operation
- We expect to have a reasonably small working set of objects that are accessed
- An object cache makes great sense



# Cache Algorithm

---

- LRU on unused objects
- Heuristic if we need to throw out a used object
  - Size
  - Cost to build the object
  - Number of subscribers
- Cache can be resized while running



# Commit

---

- Commit is done as transactions over one or several objects
- If a commit fails, the transaction is backed out and may be redone later

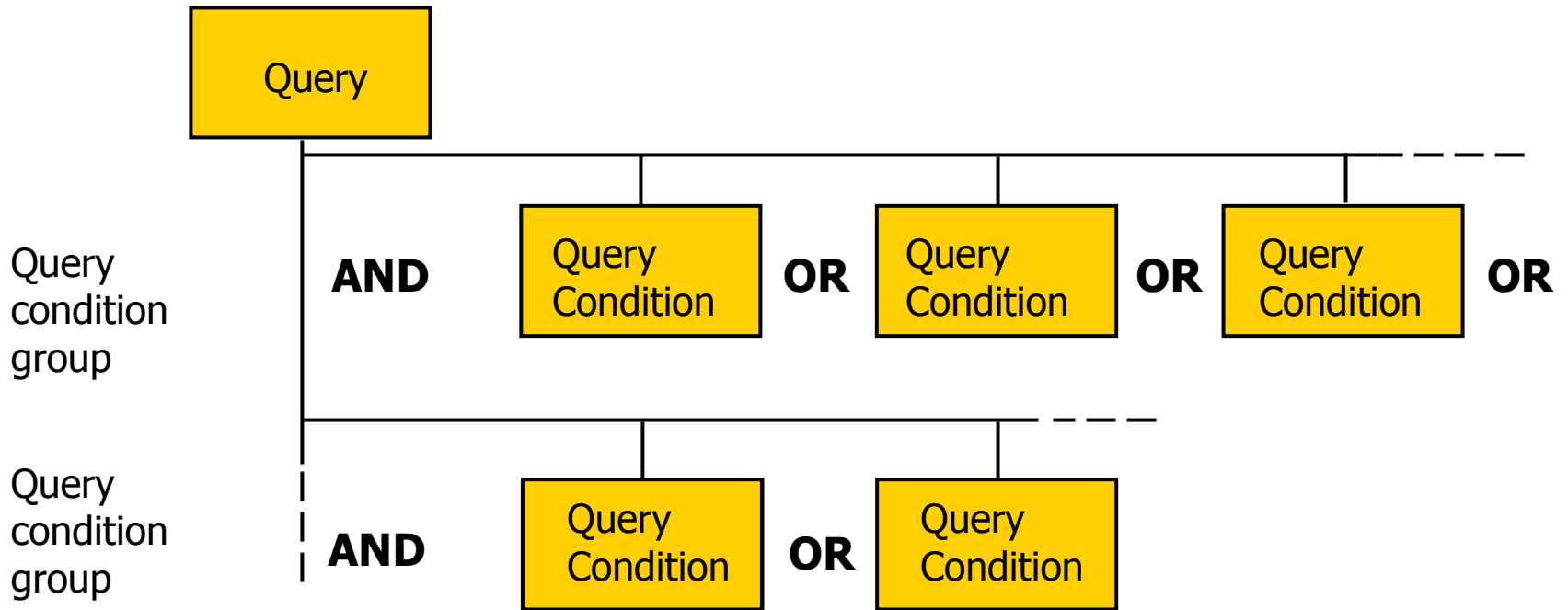


# Query object structure -> SQL query translation

---

- Client generates Python object tree
- Tree is pickled, passed to IFC and unpickled
- IFC parses tree and generates SQL

# Query Object Structure





# Subscriptions

---

- **Subscription Query**
  - A query result set is initially delivered
  - Changes to the result set are delivered as modifications to objects make them join or leave the result set
- **Subscription Request**
  - A requested object is initially delivered
  - Updates are delivered whenever the object changes
- **Non-sub queries & requests also work**



# Events

---

- Something needs to be done as a result of the world changing
- Plain events
- Time events
- Rule based events
  - On object creation



# Event Structure

---

- Transition expressions
- IDs of objects involved
- Issuing Business Logic
- Business Logic Module
- User
- Action to be taken



# Event Context

---

- Evaluation in the context of a user
  - Attribute access limitations
- Execution in the context of a user
  - IFC needs to know that there is a BL (or other advanced piece of logic) and must be able to submit pieces of code for execution
  - BL, BLM, User define the context



# Result

---

- The IFC is currently very stable and reasonably fast
- It will work for applications with soft real-time behaviour
- It is expected to handle thousands of concurrent users
- It provides a very powerful set of services with a very lean API



# Future Directions

---

- Timesharing
- Optimisation
- Concurrency
- Redundancy
- Query Caching
- Point in time access



# Contact

---

- <http://www.strakt.com>
- [info@strakt.com](mailto:info@strakt.com)
- [jacob@strakt.com](mailto:jacob@strakt.com)