Database Integration

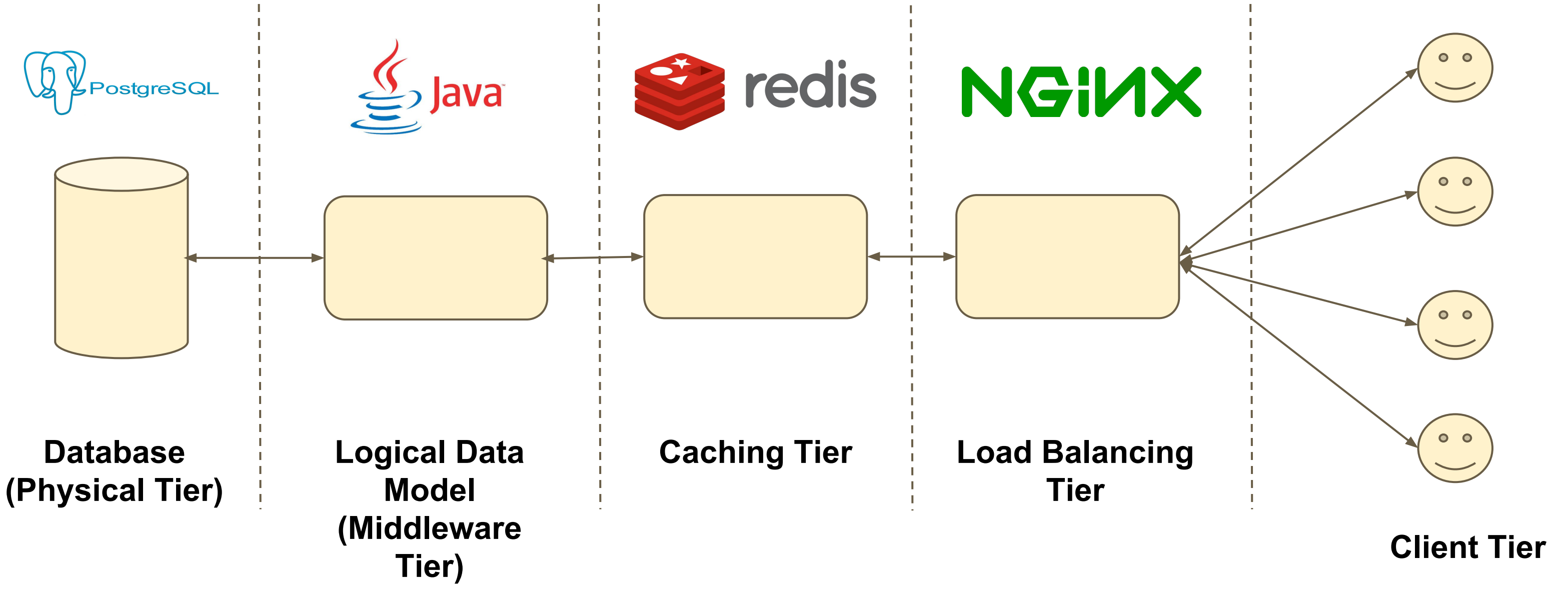
CMPU4023 - Enterprise Application Development

Database Connectivity

- A typical enterprise database will support the relational modeling of data and an SQL data query language interface
- While it is technically possible to create client applications which directly consume from the database using SQL it is more often the case that the database relational model it abstracted into a logical data model (LDM)
- The LDM is then typically implemented in a middleware tier implemented in an object-oriented language such as Java, Python or Ruby
- The middleware abstracts the connectivity details of the database which is specific to a database vendor

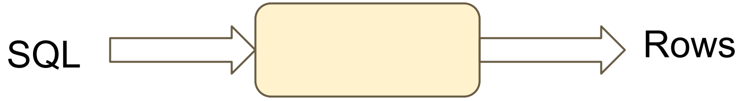
Service Architecture

- The logical view of an enterprise service architecture, based on web stack components, looks like something like this:



Query Execution

- A modern RDBMS is a very sophisticated application, heavily optimised for relational data processing - four decades of R&D
- A component known as a query planner parses SQL statements and decides the most optimal execution strategy including how to scan tables, what indexes to use, how to order query predicates, what to cache etc
- This sophistication is largely hidden from the query client which can treat the database as a black box



- The DBA is charged with maintaining the most optimal schema organisation for the query and update workload

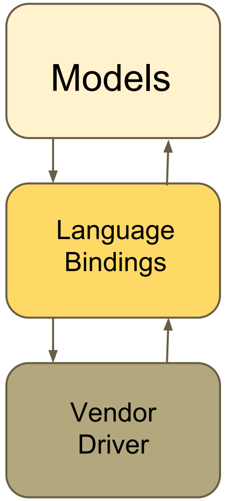
Logical Data Layer

- The logical data layer abstracts the database logic and entities in to some language-specific models (e.g. classes and objects)
- In theory, the logical data layer should be entirely independent of the database vendor implementation
- The enterprise developer's should be hidden from the details of how the database accepts queries, executes them, returns results and the specifics of how data is represented in the database
- To achieve this, the logical data layer itself is made up of a number of internal logical layers in turn

Inside the LDM (RDBMS)

- The RDBMS LDM comprises typically three internal layers of abstraction

- **Models** - Language-specific data structures which abstract the representation of the database entities (e.g. classes and objects)
- **Language bindings for SQL** - Generic API for interface from the middleware language to SQL services on the database (e.g. execution of queries)
- **Database vendor driver** - Implementation-specific logic for translating SQL bindings into the raw connection API (e.g. binary protocol for parameterised query data)



Models

- Models are the language-specific abstraction of the database entities
- In RESTful API services, models provide the architectural bridging between database entities and the exported resource state and behaviour
- In the simplest view, a model is just an internal memory representation of persistent data structures which can be validated, processed and represented independently of the database itself
- In practice, models are implemented as classes and objects in OOP languages like Java

SQL Language Bindings

- A given language such as Java or Python will provide generic API facilities for interfacing with SQL on a target RDBMS
- This API allows for the standardised representations of such things as SQL query strings, query parameters, results (e.g. rows) and error codes
- Models are built directly or indirectly on top of these SQL API bindings - a subject we'll discuss in more detail when considering object-relational mappers

Vendor Driver

- The driver layer abstracts the vendor-specific details of the database interface hiding details such as:

  - The binary wire communications protocol and connection management
  - Establishing user access credentials and connectivity security
  - Connection pool management
  - Thread-safety
  - Query string and parameter representation
  - Results representation and error codes

- In most cases, one RDBMS driver can be transparently substituted for another [RDBMS driver] with affecting the semantics of the logical data model

Alternative Database Architectures

- The so-called NoSQL family of databases are beginning to see some use in enterprise albeit to a much lesser extent than traditional RDBMS
- Much of the same abstractions and layering would exist in such scenarios which the obvious observation the the SQL abstraction layer of the logical data model would not be necessary
- NoSQL is not yet a standard so models are often built directly on top of the database drivers at the expensive of tighter coupling between them and the underlying database representation
- NoSQL solutions tend to be harder to swap out as a result

Summary

- An enterprise service stack potentially comprises many separate interconnecting tiers including a database tier and a middleware tier
- The middleware tier is responsible for implementing the logical data model which is an abstraction of the database entities, relationships and data representation
- The LDM itself comprises a model layer, a SQL binding layer and a database driver
- The application programmer view of the database is provided by a simpler model-based abstraction of the underlying complexity