



# **ATLAS Online Software**

ConditionsDB MySQL  
Backend  
Implementation

*PROGRESS INDICATOR*

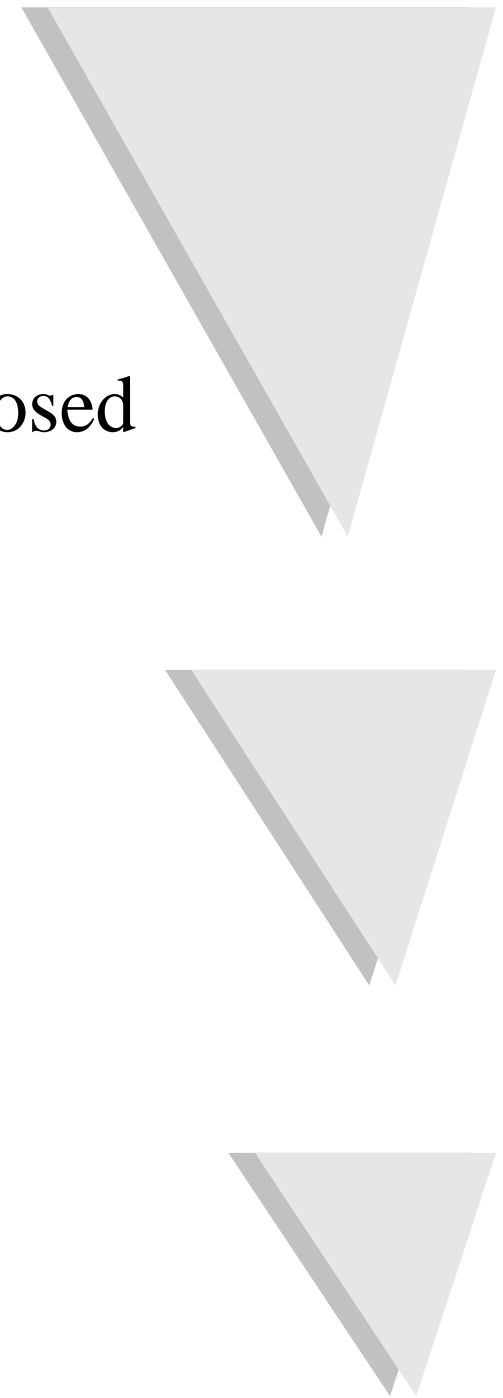


# API layered structure

**Upper layer:** set of virtual classes imposed by the interface specification

**Middle layer:** implementation specific concrete classes, derived from the interface classes.

**Bottom layer (NEW) :** mySQL tight connected classes (replacing the old implementation's wrapper functions)



# ?? Data Constraints ??

Upper layer: the overall database structure is a guaranteed at this level

Middle layer: specific topics like the “*time validity range*”, the “*insertion time*”, the correctness of folderset/folder hierarchy are assured by this layer.

Bottom layer: Time partitioning policies and correctness

MySQL level: *define default values... what else?*

# Why the additional bottom layer

Not quite a new layer, but a replacement for a set of wrapper C like functions.

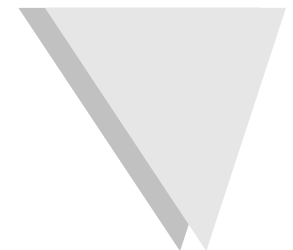
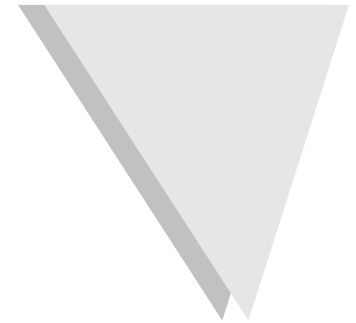
Allows one to plug/unplug/replace new features not foreseen in the interface:

Time partitioning management.

Clearer code interfaces:

Code is easier to maintain or extend;  
improved robustness.

Painless integration with the rest of the code and possibly more efficient coding (and code).



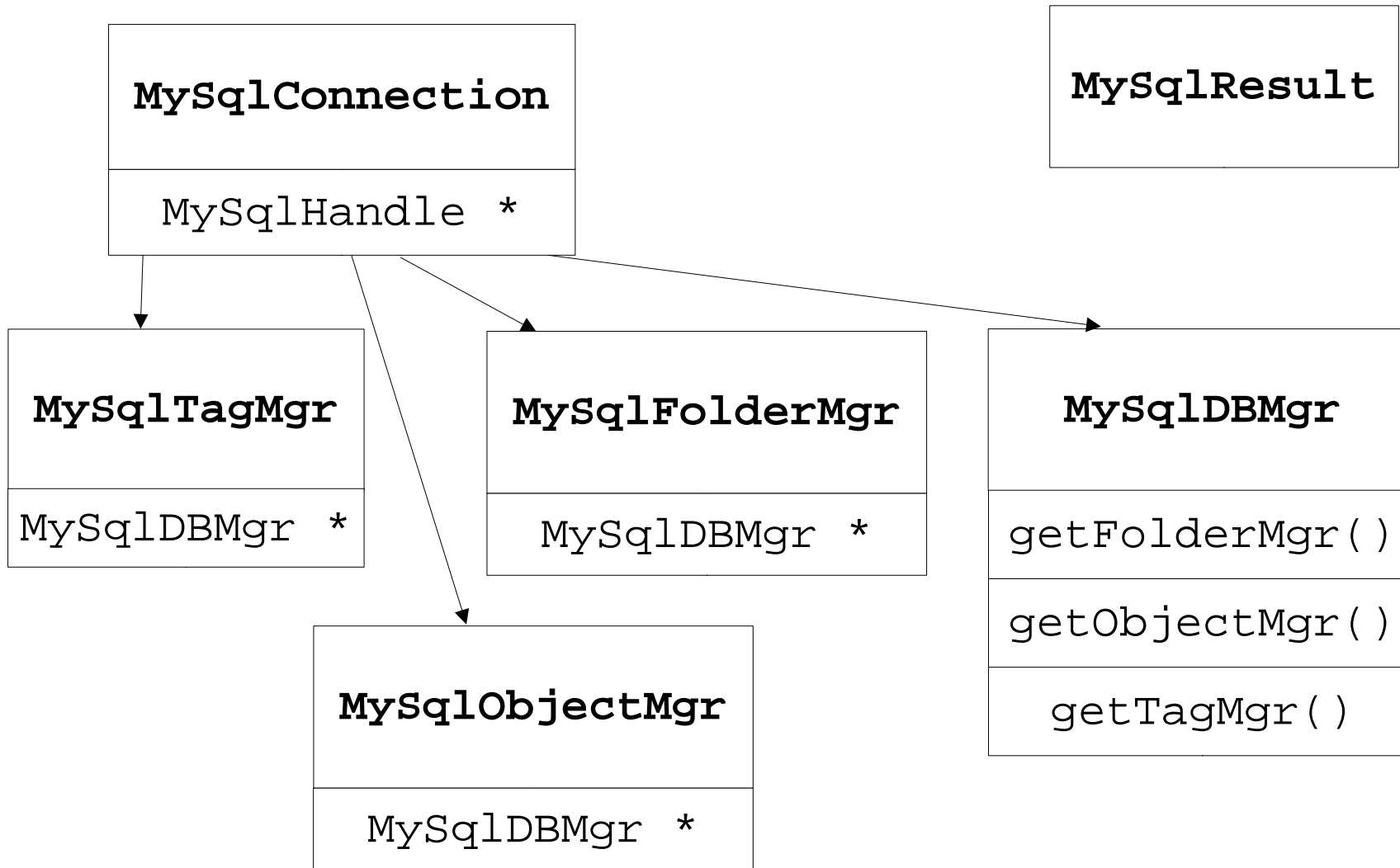
# PRE-RELEASE NOTES

Use LIMIT SQL keyword to retrieve only the desired rows instead of using `mysql_use_result()` to avoid retrieving a large number of rows.

The `mysql_use_result()` has many drawbacks in the implementation when compared to `mysql_store_result()`.

Use of `strstreams` to build up the queries and the `mysql_real_query()`, instead of `mysql_query()`, provide a more efficient approach specially when dealing with large queries.

# *Bottom layer snapshot*



# Architecture

