Metadata in the TAP context (1)

The Problem:

- learn about which tables, tablesets, ... are available from a TAP server
- for each of the tables / tablesets, discover their contents:
  - descriptions, characterisation, keywords
  - relations between tables *(keys)*
  - statistics (rows, region covered)
  - column contents: description, units, utypes, UCDs, datatypes, domains, ...
- part or all is known / cached in Registries

Metadata in the TAP context (2)

What are metadata ? Essentially 3 parts:

- semantics *(science metadata)*: describes role of tables *(e.g. observational data, model results, etc)* and column contents *(units, detailed explanations, UCDs, utypes, references...)*
- logistics *(implementation metadata)*: organisation, datatypes, primary/foreign keys, unicity, domain...
- statistics *(statistical properties)*: summary of table(set)s like row counts, sky regions for source catalogs, column min/max/σ, ...

Why a metadata access ?

- Find out table(set)s that could match my requirements. Typical examples:
  - which table(set)s contain redshifts in a given region of the sky ? *(statistics)*
  - which table delivers calibrated fluxes in some IR band *(semantics)*
- Some parts *(almost all?)* exist in Registries – is it necessary to access metadata at the TAP server ?
  YES because:
  - more up-to-date (delay of propagation to Registries)
  - metadata generally richer than what's in the Registry

How to access metadata ?

Essentially two possible ways:

- by *methods* (functions) which return some IVOA-defined object (e.g. VOResource objects or identifiers). Several methods may exist: getResource(), keywordSearch(), ...
- by a set of (system) tables similar to what is defined in the INFORMATION_SCHEMA
  - can be queried like other tables
  - can accept complex constraints

INFORMATION_SCHEMA (1)

- is a standardized *system tableset*
- introduced in SQL92
- typical contents:
  - information_schema.schemata = list of tablesets
  - information_schema.tables = list of tables
  - information_schema.columns = list of columns
  - information_views = list of views (stored queries)
  - etc...

INFORMATION_SCHEMA (2)

BUT

- does not exist in all relational DBs
- not identical among different DBs
- is aimed to describe DB schema only (tables, columns, keys, protections, ...), i.e. *logistics*
- most important *(semantics)* parameters are missing! For example:
  - no notion of units, UCDs, etc... in columns
  - no notion of date of last modification in tables

INFORMATION_SCHEMA (3)

Therefore...

- information_schema alone can't be the solution!
- alternatives:
  - definition of some IVOA standard set of “meta” tables containing the required 3 sets of metadata, similar to what exists in VizieR *(METAcat, METAtab, METAcol)*
  - define tables which complement what exists in information_schema *(many joins!)*
  - ... keep just the methods ...

Metadata Query Output

Result of a search for tables satisfying some constraint may be returned in several forms:

- a list of tuples of type information_schema.tables (or IVOA-defined *table of tables*) expressed as a VOTable (each row describes one table)
- a set of VOResource identifiers or objects (limited set of metadata!)
- empty VOTable(s) with/without all FIELD(s)