SSA Primary Spectrum Table (DALServer Framework)					
Column Name	**Datatype**	**Constraint**	**Value**	**Example**	**Description**
id	integer	not null	primary key		Spectrum table primary key
obs_id	varchar(32)	not null	key	vla_12345	Observation identifier
archive_id	varchar(128)	not null	key or path	jvla/w50_c+d-15arcsec.fits	Identity of dataset in storage subsystem
preview_id	varchar(128)		key or path	jvla/w50_c+d-15arcsec.jpeg	Identity of preview in storage subsystem
htm_id	integer		HTM	2151573366	Hierarchical triangular mesh spatial index
access_format	varchar(32)	not null	MIME type	text/xml;x-votable	MIME type of dataset serialization
access_estsize	integer	not null	kB		Estimated or approximate dataset size in kB
dataproduct_type	varchar(16)	not null	obstap	Spectrum	Primary dataset type as defined by ObsCore
dataproduct_subtype	varchar(32)	not null	obstap	gbt-sdfits	Dataset type within local archive
calib_level	integer	not null	obstap	2	Dataset calibration level
dataset_length	integer	not null	pixels	4096	Number of samples in spectrum
spec_spectral_axis	varchar(16)			freq	Table spectral coordinate column name
spec_flux_axis	varchar(16)			flux	Table flux column name
obs_title	varchar(128)	not null		VLA U-band 3C273 F300W 35ksec	Spectrum title briefly describing Spectrum content
obs_creator_name	varchar(32)	not null		VLA_Pipeline	Name of entity that created the dataset
obs_collection	varchar(32)	not null		VLA	Data collection name
obs_creator_did	varchar(160)		IVO datasetID	ivo://nrao/vla#12345	Creator-specified dataset identifier
obs_publisher_did	varchar(160)	not null	IVO datasetID	ivo://nrao/archive#vla-12345	Publisher-specified dataset identifier
obs_dataset_did	varchar(160)		ADS datasetID	ADS/NRAO.VLA#12345	ADS (or comparable) dataset identifier
obs_release_date	varchar(20)		ISO8601	2014-09-22	Date dataset was/will be publically released
obs_creation_date	varchar(20)		ISO8601	2013-09-22	Date dataset was created
facility_name	varchar(20)	not null		NRAO	Name of facility that created the dataset
instrument_name	varchar(20)	not null		VLA	Instrument used to generate the data
obs_bandpass	varchar(20)			U	Observed bandpass
obs_datasource	varchar(20)	not null		pointed	Source of the data (pointed, survey, theory, etc.)
proposal_id	varchar(20)			P_12345	Observing project proposal identifier
target_name	varchar(20)			3C273	Target name if any
target_class	varchar(20)			quasar	Target classification
s_ra	double	not null	ICRS	187.2792	Center of field / Spectrum
s_dec	double	not null	ICRS	2.0525	Center of field / Spectrum
s_fov	double	not null	deg		Field of view of observation
s_region	varchar(128)		AstroCoordArea	circle icrs 11.34 -20.56 0.083	Footprint of observation / Spectrum
s_calib_status	integer			absolute	Level of spatial calibration
s_resolution	double		arcsec		Spatial resolution (observed signal, not detector)
em_min	double	not null	m		Spectral bandpass, lower limit
em_max	double	not null	m		Spectral bandpass, upper limit
em_resolution	double		m		Spectral resolution (characteristic value)
em_respower	double				Spectral resolving power (characteristic value)
t_min	double	not null	mjd		Temporal bandpass, lower limit
t_max	double	not null	mjd		Temporal bandpass, upper limit
t_exptime	double		s		Time resolution
t_resolution	double		s		
o_ucd	varchar(20)			phot.flux;em.radio.200-400MHz"	UCD for observable
o_unit	varchar(20)	not null		jy/beam	Unit for observable
pol_states	varchar(20)			I Q U V	Polarization states represented in dataset

Notes	
Data Models	The IVOA Spectrum data model contains the core of the Observation data model (ObsCore) as a subset, hence much of what is defined here is from ObsCore; refer to the ObsTAP specification for details on these standard fields. Fields with the "spec_" prefix are specific to the Spectrum data model. Some other fields, e.g., ID, ARCHIVE_ID, HTM_ID, etc., are specific to the DALServer implementation. A service query response such as for SSA may return additional metadata not shown here, as some query response fields are generated by the service from other Spectrum table metadata.
Null Values	All table columns defined here must be physically present in the database table to avoid illegal SQL queries, however a number of fields are permitted to have null values. The Spectrum table is not quite as loose with null values as ObsCore as fewer fields are allowed to have null values, since we are dealing with a specific type of data with more well-defined characteristics (ObsCore has to be able to represent any science data product).
Metadata Extension	The Spectrum table may be extended by adding custom metadata specific to the collection or collections described by the table. Depending upon the capabilties of the Spectrum service, this extra metadata may or may not be passed-through for display in a client application, or be available for use as additional query constraints in a discovery query. In particular, each SSA service instance has a distinct corresponding primary Spectrum table used to drive the service. Large Spectrum data collections should usually be served by their own Spectrum service, in which case one can add additional metadata to the Spectrum table specific to the Spectrum data collection being served. A processing pipeline or survey for example, will usually define some standard metadata specific to the instrument, pipeline, or survey, which will be useful to pass through to clients or possibly be used to refine a discovery query.
Normalization	To simplify and optimize queries, the Spectrum table is a simple flat table (as is the ObsTAP table). Much of the metadata describing data products is in common with the ObsTAP table, if both are present, i.e., some metadata is duplicated in both tables. To avoid duplication of information in the underlying DBMS, one will normally want to store the fundamental metadata in lower level, fully normalized tables, and produce the Spectrum and ObsTAP index tables in some automated fashion, updating them as data is added or other changes are made to the underlying DBMS tables; for a simple static data collection the Spectrum table can be produced once and left alone. Runtime access to these tables by VO services is read-only.
id	ID is what is used externally (e.g., in access reference URLs) to refer to an Spectrum. The Spectrum table tablename and the ID of the Spectrum dataset within that table uniquely specify the Spectrum without exposing details such as the internal file pathname specifying where the Spectrum is stored.
obs_id	OBS_ID uniquely identifies an "observation" within the context of the Spectrum table, or within the context of a single instrument. A typical example might be the instrument name concatenated with a running number, time of observation, or some other sequence. If multiple data products belong to the same observation they all share the same OBS_ID. What constitutes an "observation" is instrument-specific, but it usually refers to any data collected within a given time frame by a specific instrument configuration, as defined by the given metadata.
archive_id	ARCHIVE_ID uniquely identifies the data product within the storage subsystem used to store data for the data collection. In the simplest case this can be just the file pathname of the data product within the storage subsystem, e.g., relative to some root directory, as specified in the service configuration. In a more complex case, ARCHIVE_ID is merely some unique key used to identify a particular data product within the storage subystem. This enhances storage virtualization, making it possible to relocate data products, maintain replicated data, etc., transparently to other elements of the archive system. Whether ARCHIVE_ID is used for direct file access by pathname, or indirect access by key, is specified when the service is configured.
preview_id	PREVIEW_ID is the archive_id of the preview graphic (if any) for the data product. This should be a modest-size graphic (e.g., jpeg) rendition of the data product, suitable for scaled-down display in a query response table. Higher resolution graphic Spectrums should be represented as separate Spectrum data products in the main Spectrum table. Related data products such as a FITS Spectrum and a preview form an association of some sort, "observation" being a primary example of an association type. [An alternative approach to storing preview graphics is to store them as a blob directly in the Spectrum or obstap table, if the service implementation supports that option.]
dataproduct_type	The primary data product type as defined in ObsCore ("Spectrum", "cube", "spectrum", and so forth). For multi-dimensional Spectrum data, a 2-D Spectrum is of type "Spectrum", and an n-D Spectrum is of type "cube" if n>2. An extracted spectrum, if represented as a one-dimensional array, is of type "spectrum", with the file format specifying how the spectrum is represented, e.g., as an IVOA Spectrum object or a FITS 1D Spectrum. A visibility dataset has the type "visibility". Visibility datasets can be Spectrumd on the fly by a sufficiently capable Spectrum service hence could be included in the primary Spectrum table by an advanced service. Likewise, an X-ray event list dataset, of type "event", could be considered a specialized type of multi-dimensional Spectrum.

dataproduct_subtype	The data product subtype is finer-grained than dataproduct_type, defining the type of data product within a specific archive or data collection. The subtype should specify the *logical* type of the data product. For example, if the standard data processing pipeline for an instrument produces N data products specific to the instrumental data for a single observation, each should be assigned a separate subtype and all would share the same OBS_ID. If this set of data products is also available packaged as a single gzipped tar file (or whatever), then the "package" data product would have a separate subtype indicating that it contains the full package of data products for an observation. Both the entire observation package and selected individual data products, each with an assigned subtype, could be simultaneously exposed in something like an ObsTAP query response. Data product subtypes should be in the form of simple keys suitable for use as query constraints in a database query.
spec_spectral_axis	If the spectrum is serialized as a table, *spec_spectral_axis* specifies the name of the column contaiing the spectral coordinate. Likewise, *spec_flux_axis* specifies the name of the table column containing the flux vector.