Eleanor Schema Documentation

Table of contents

Overview

The primary function of **Eleanor** is to accept a problem detailing how to select fully-specified aqueous speciation problems, referred to as an "order", equilibrate them, and curate the resulting data.

Each such selected problem corresponds to a point (row) in the "variable space" (`variable_space` table). Each order may then have any number of associated variable space rows. The properties of the variable space points are broken out over a number of tables, e.g. the `elements` table contains the elemental composition of the fluid before speciation, the `mineral_reactants` table contains the amount and titration rate of each mineral to be reacted with the system, etc…

Speciation is then handled by the kernel specified in the order, which may produce zero or more points (rows) in the "equilibrium space" (`equilibrium_space` table). Each row for a given variable space point represents some form or degree of equilibration. For example, the `eq36` kernel performs the equilibration in two stages, first speciating the fluid alone (the `"eq3"` stage) and then equilibrating the fluid with the reactants (the `"eq6"` stage). As such, any variable space point equilibrated with the `eq36` kernel will have at least two equilibrium points associated with it (more on this below). As with variable space points, several tables contain detailed information about the equilibrium space points, e.g. the `equilibrium_elements` table contains the elemental composition of the fluid after the given speciation, the `equilibrium_reactants` table contains information about how much of each reactant has been, and how much remains to be reacted, etc…

Tables

This document describes each table in the schema and its relationship to the other tables. For brevity we use a few abbreviations as follows:

Key	
PK	Primary Key
FK	Foreign Key
N	Nullable

Every table a has single primary key (**PK**)—`id`—without exception, though that my change in subsequent versions of **Eleanor** (but probably not).

As described in Overview, the **Eleanor** database is somewhat hierarchical with the `orders` table as the root, zero or more entries in the `variable_space` table referring to an entry in `orders`, and with zero or more entries in the `equilibrium_space` table referring to an entry in the `variable_space` table. These relationships are reflected in two ways in this document:

1. The document headings are structured such that each table refers to the table one-level above it.
2. The foreign keys (**FK**) are noted in the tables, and the associated column name links to the referenced table. These column names are generally of the form `*_id`. The only exceptions to this rule are for tables with a one-to-one, rather than many-to-one, relationship, e.g. there is at most one `huffer` entry for each `orders`, so the **FK** is simply `id`.

Most columns in the **Eleanor** database are `NOT NULL`, meaning they are guaranteed to have a value, though that value may be some standard value representing "absent" or "missing", e.g. `-inf`. Columns that are nullable (**N**) are denoted as such.

orders

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The order identifier	
			`name`	`VARCHAR`	The name of the order	
			`hash`	`VARCHAR`	A SHA-256 hash of the order file contents	
			`eleanor_version`	`VARCHAR`	The version of eleanor used to run the order	
			`raw`	`JSON`	The raw, JSON-encoded contents of the order	
			`create_date`	`DATETIME`	The datetime at which the order was created	

When the user runs an order, **Eleanor** stores an entry in the `orders` table which includes the autoincremented `id`, the name of the order (extracted from the YAML/TOML/JSON file), a SHA-256 hash of the normalized content of the order, and the version of **Eleanor** used to execute the order.

A entry in the `orders` table is created if:

1. The hash of the order is not in the table (i.e. a new order), or
2. The version of **Eleanor** used to run the order changes, or
3. The user specifies that they want a disctinct order created.

Otherwise, any variable space points generated for the order will be associated with an existing order with the same `hash` and `eleanor_version`. This allows the user to run small batches for initial testing and subsequent batches to more fully sample the variable space, an "extension run".

The `create_date` value is the datetime at which the order was initially created. Subsequent extension runs do not update the `create_date`.

Example:

```
SELECT id, name, hash, eleanor_version, create_date FROM orders;

+----+----------------------+------------------------------------------------------------------+-----------------+-------------------------+
| id | name                 | hash                                                             | eleanor_version | create_date             |
|----+----------------------+------------------------------------------------------------------+-----------------+-------------------------|
| 1  | A Wild Example Order | 9a511f8f977d86264a8329b8a75bc9d489237d6ad249ab009c73f8e159360c36 | 0.19.0          | 2026-01-09 10:59:01.98 |
+----+----------------------+------------------------------------------------------------------+-----------------+-------------------------+
```

huffer

PK	FK	N	Column Name	Type	Description	Unit
✓	✓		id	INTEGER	The huffer entry identifier corresponds to the order id	
			exit_code	INTEGER	The exit code generated by the huffer (non-zero for failure)	
			zip	BLOB	A zip archive of the input files used and output files generated when running the huffer. This includes a traceback when an error occurs	

The user may opt to run the `huffer` to check the validity of the order they have submitted before generating variable space data. The huffer chooses a single variable space point consistent with the order and fully speciates it. The exit code generated by the kernel (non-zero for failure) is stored along with a `zip` archive of the kernel input and output files and a traceback if an error occurred.

This table is primarily useful for debugging.

Example:

```
SELECT * FROM huffer;

+----+-----------+----------+
| id | exit_code | zip      |
|----+-----------|----------|
| 1  | 0         | <binary> |
+----+-----------+----------+
```

variable_space

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The variable space point identifier	
	✓		order_id	INTEGER	The order id for which the point was generated	
			temperature	DOUBLE	The temperature	$^{\circ}C$
			pressure	DOUBLE	The pressure	bar
			exit_code	INTEGER	The exit code generated by the kernel (non-zero for failure)	
			create_date	DATETIME	The datetime when the variable space point was created	
			start_date	DATETIME	The datetime when the equilibration started	
			complete_date	DATETIME	The datetime when the equilibration completed	

When the user runs **Eleanor**, a specified number of variable space points are selected based on the constraints provided by that order. Each point represents the unspeciated initial state of a system and includes such information as:

- Temperature
- Pressure
- Kernel Settings (kernel)
- Elemental Composition (elements)
- Species Constraints (species)
- Reactant Specifications, including:
    - Aqueous Reactants (aqueous_reactants)
    - Element Reactants (element_reactants)
    - Fixed Gas Reactants (fixed_gas_reactants)
    - Gas Reactants (gas_reactants)
    - Mineral Reactants (mineral_reactants)
    - Solid Solution Reactants (solid_solution_reactants)
    - Special Reactants (special_reactants)
- Suppressions (suppressions)

The variable_space table stores the core information about the variable space point per row. This includes an autoincremented id, the temperature and pressure of the system, and the datetime at which it was generated. Each point refers back to the order to which it belongs via the order_id.

After generating the variable space point, **Eleanor** proceeds to speciate the system using the user-specified kernel. The exit_code generated by the kernel as well as the datetimes of when the speciation started and completed are stored in the variable_space table as well.

Example:

```
SELECT * FROM variable_space ORDER BY id LIMIT 1;

+----+----------+-------------+----------+-----------+----------------------------+----------------------------+----------------------------+
| id | order_id | temperature | pressure | exit_code | create_date                | start_date                 | complete_date              |
|----+----------+-------------+----------+-----------+----------------------------+----------------------------+----------------------------|
| 1  | 1        | 300.0       | 500.0    | 0         | 2026-01-09 10:59:02.541519 | 2026-01-09 10:59:02.546294 | 2026-01-09 10:59:02.843071 |
+----+----------+-------------+----------+-----------+----------------------------+----------------------------+----------------------------+
```

scratch

PK	FK	N	Column Name	Type	Description	Unit
✓	✓		id	INTEGER	The scratch entry identifier corresponds to the variable space id	
			zip	BLOB	A zip archive of the input files used and output files generated when equilibrating the variable space point. This includes a traceback when an error occurs	

The input files used and output files generated by the kernel when speciating a "variable space" point are stored in a zip archive in the scratch table. The scratch row will have the same id as the associated variable space point.

By default, **Eleanor** only stores scratch entries if an error occurs while speciating the variable space point, as signified by a non-zero exit code from the kernel. In such a case a `traceback.txt` file is included in the `zip` archive detailing the error. The user may specify, at runtime, that scratch be collected for all variable space speciations. In that case, no `traceback.txt` file is included in the `zip` archive.

Example:

```
SELECT * FROM scratch WHERE variable_space_id = 1;

+----+----------+
| id | zip      |
|----+----------|
| 1  | <binary> |
+----+----------+
```

kernel

PK	FK	N	Column Name	Type	Description	Unit
✓	✓		id	INTEGER	The kernel entry identifier corresponds to the variable space point	
			type	VARCHAR	The type of the kernel used, e.g. `eq36`	
			settings	JSON	A JSON-encoded object of the settings passed to the kernel	

Eleanor is designed to support user-provided kernels which may require specific configuration or settings. Those settings are stored in the `kernel` table, one row for each variable space point. The kernel settings row will have the same `id` as the variable space point to which it belongs. The `type` of the kernel is stored alongside a JSON encodedded `settings` object, the form of which is kernel-specific.

While, at the moment, only the `eleanor.kernel.eq36` kernel is provided out-of-the-box, future versions may include other kernel implentations or the user may design and implement their own. It is then difficult to design a schema for storing the settings. Using a JSON object makes things simpler given PostgreSQL's native JSON support.

Example:

```
SELECT * FROM kernel WHERE id = 1;

+----+------+-------------------------------------------------------+
| id | type | settings                                              |
|----+------+-------------------------------------------------------|
| 1  | eq36 | {"model": 0, "timeout": null, "basis_map": {}, ... } |
+----+------+-------------------------------------------------------+
```

elements

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The element identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which elemental molality was generated	
			name	VARCHAR	The name of the element (atomic symbol)	
			log_molality	DOUBLE	The log molality of the element in the variable space solution	$\log molal$

The elemental composition of the system's fluid must be provided as part of the specification of a variable space point. Each element in the fluid is stored in the `elements` table with a unique, autoincremented `id`, the name of the element, and the `log_molality` that was chosen when the variable space was generated. Each element references the variable space point it is associated with by the `variable_space_id`.

Example:

```
SELECT *
 FROM elements
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+------+--------------+
| id | variable_space_id | name | log_molality |
|----+-------------------+------+--------------|
| 1  | 1                 | C    | -9.0         |
| 4  | 1                 | Ca   | -9.0         |
| 5  | 1                 | Cl   | -9.0         |
| 7  | 1                 | Fe   | -9.0         |
| 6  | 1                 | Mg   | -9.0         |
| 8  | 1                 | N    | -9.0         |
| 2  | 1                 | Na   | -9.0         |
| 3  | 1                 | Si   | -9.0         |
+----+-------------------+------+--------------+
```

species

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The species identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the species molality/activity/fugacity was generated	
			name	VARCHAR	The name of the species	
			value	DOUBLE	The log molality/activity/fugacity of the species in the variable space solution	variable

Constraints on the aqueous and gaceous species in the system may, though in some cases must, be provided as part of the specification of a variable space point, such as the pH and fO_2 of the system. Each species constraint is stored in the `species` table with a unique, autoincremented `id`, the name of the species, and the `value` of the constraint (log molality, log activity or log fugacity as appropriate) that was chosen when the variable space was generated. Each element references the variable space point it is associated with by the `variable_space_id`.

Example:

```
SELECT *
 FROM species
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+-------+-------+
| id | variable_space_id | name  | value |
|----+-------------------+-------+-------|
| 1  | 1                 | H+    | -7.0  |
| 2  | 1                 | O2(g) | -1.0  |
+----+-------------------+-------+-------+
```

Reactants

Several types of reactant can be specified in an order, facilitating the simulation of a wide variety of systems. These types are:

- Aqueous Reactants (aqueous_reactants)
- Element Reactants (element_reactants)
- Fixed Gas Reactants (fixed_gas_reactants)
- Gas Reactants (gas_reactants)
- Mineral Reactants (mineral_reactants)
- Solid Solution Reactants (solid_solution_reactants)
- Special Reactants (special_reactants)

and each the details of reactants are stored in tables by type. However, they mostly follow the same basic form:

1. Each table has an autoincrementing `id`,
2. Each table has a `variable_space_id` that associates the entry with the variable space point,

3. The `name` of the reactant,
4. The molar amount of the reactant, `log_moles`, and
5. Either a `titration_rate` or `log_fugacity` as appropriate for the reactant type.

aqueous_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The aqueous reactant identifier, specific to the variable space point	
	✓		`variable_space_id`	`INTEGER`	The id of the variable space point for which the aqueous reactant amount and titration rate were generated	
			`name`	`VARCHAR`	The name of the aqueous reactant	
			`log_moles`	`DOUBLE`	The log amount of the reactant to be reacted with the fluid	$\log mol$
			`titration_rate`	`DOUBLE`	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

An aqueous reactant models the addition (or removal) of a fixed molar quantity (`amount`) of an aqueous species into the system at a ξ-rate (`titration_rate`).

Example:

```
SELECT *
 FROM aqueous_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+-------+-----------+----------------+
| id | variable_space_id | name  | log_moles | titration_rate |
|----+-------------------+-------+-----------+----------------|
| 1  | 1                 | CaCl2 | -2.0      | 1.0            |
+----+-------------------+-------+-----------+----------------+
```

element_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The element reactant identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the element reactant amount and titration rate were generated	
			name	VARCHAR	The name of the element reactant	
			log_moles	DOUBLE	The log amount of the reactant to be reacted with the fluid	$\log mol$
			titration_rate	DOUBLE	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

An element reactant is just a special case of special_reactant in which a fixed molar quantity (amount) of a single (unionized) element is added to the bulk composition of the fluid at some ξ-rate (titration_rate).

Example:

```
SELECT *
 FROM element_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+------+-----------+----------------+
| id | variable_space_id | name | log_moles | titration_rate |
|----+-------------------+------+-----------+----------------|
| 1  | 1                 | Si   | -2.0      | 1.0            |
+----+-------------------+------+-----------+----------------+
```

fixed_gas_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The fixed gas reactant identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the fixed gas reactant amount and fugacity were generated	
			name	VARCHAR	The name of the fixed gas reactant	
			log_moles	DOUBLE	The log amount of the fixed gas buffer	$\log mol$
			log_fugacity	DOUBLE	The log fugacity of will be titrated into the fluid	mol/ξ

A fixed gas reactant models gas exchange with a large, external gas reservoir at a fixed fugacity. Rather than the gas being "titrated" in, the log_fugacity is enforced through mass exchange with a buffer with initial size amount.

Example:

```
SELECT *
 FROM fixed_gas_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+--------+-----------+--------------+
| id | variable_space_id | name   | log_moles | log_fugacity |
|----+-------------------+--------+-----------+--------------|
| 1  | 1                 | CO2(g) | -0.3      | -3.5         |
+----+-------------------+--------+-----------+--------------+
```

gas_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The gas reactant identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the gas reactant amount and titration rate were generated	
			name	VARCHAR	The name of the gas reactant	
			log_moles	DOUBLE	The log amount of the reactant to be reacted with the fluid	$\log mol$
			titration_rate	DOUBLE	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

A gas reactant models reacting a fixed molar quantity (`amount`) of a gaseous species with the fluid at some ξ-rate (`titration_rate`).

Example:

```
SELECT *
 FROM gas_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+-------+-----------+----------------+
| id | variable_space_id | name  | log_moles | titration_rate |
|----+-------------------+-------+-----------+----------------|
| 1  | 1                 | N2(g) | -0.3      | 1.0            |
+----+-------------------+-------+-----------+----------------+
```

mineral_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The mineral reactant identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the mineral reactant amount and titration rate were generated	
			name	VARCHAR	The name of the mineral reactant	
			log_moles	DOUBLE	The log amount of the reactant to be reacted with the fluid	$\log mol$
			titration_rate	DOUBLE	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

A mineral reactant models reacting a fixed molar quantity (`amount`) of a pure mineral with the fluid at some ξ-rate (`titration_rate`).

Example:

```
SELECT *
 FROM mineral_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+----------+-----------+----------------+
| id | variable_space_id | name     | log_moles | titration_rate |
|----+-------------------+----------+-----------+----------------|
| 1  | 1                 | hematite | -0.3      | 1.0            |
+----+-------------------+----------+-----------+----------------+
```

solid_solution_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The solid solution identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the solid solution amount and titration rate were generated	
			name	VARCHAR	The name of the solid solution	
			log_moles	DOUBLE	The log amount of the reactant to be reacted with the fluid	$\log mol$
			titration_rate	DOUBLE	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

A solid solution reactant models reacting a fixed molar quantity (`amount`) of a solid solution with the fluid at some ξ-rate (`titration_rate`). Each such solid solution must include the mole-fraction of each of its end members. The end member data is stored separately in the `solid_solution_reactant_end_members` table.

Example:

```
SELECT *
 FROM solid_solution_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | olivine-ss | -0.3      | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

solid_solution_reactant_end_members

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The solid solution end member id, specific to the solid solution	
	✓		solid_solution_reactant_id	INTEGER	The id of the solid solution for which this is an end member	
			name	VARCHAR	The name of the end member	
			fraction	DOUBLE	The mole-fraction of the end member in the solid solution	unitless

- `CONSTRAINT fraction_in_range CHECK (0.0 <= fraction AND fraction <= 1.0)`

Each solid solution reactant has a fixed distribution of end members, represented by the mole-fraction of the total solid solution.

Example:

```
SELECT em.*
 FROM solid_solution_reactant_end_members AS em
 JOIN solid_solution_reactants AS ssr ON ssr.id = em.solid_solution_reactant_id AND variable_space_id = 1
 ORDER BY em.name;

+----+-----------------------------+------------+----------+
| id | solid_solution_reactant_id | name       | fraction |
|----+-----------------------------+------------+----------|
| 1  | 1                           | fayalite   | 0.6      |
| 2  | 1                           | forsterite | 0.4      |
+----+-----------------------------+------------+----------+
```

special_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The special reactant identifier, specific to the variable space point	
	✓		variable_space_id	INTEGER	The id of the variable space point for which the special reactant amount and titration rate were generated	
			name	VARCHAR	The name of the special reactant	
			log_moles	DOUBLE	The log amount of the reactant to be reacted with the fluid	$\log mol$
			titration_rate	DOUBLE	The rate (in ξ) at which the reactant will be titrated into the fluid	mol/ξ

A special reactant models reacting a fixed molar quantity (`amount`) of a user-specified molecular composition with the fluid at some ξ-rate (`titration_rate`). Each special reactant must have include its stoichiometric composition, which is stored separately in the `special_reactant_compositions` table.

Example:

```
SELECT *
 FROM special_reactants
 WHERE variable_space_id = 1
 ORDER BY name;

+----+-------------------+------------------+-----------+----------------+
| id | variable_space_id | name             | log_moles | titration_rate |
|----+-------------------+------------------+-----------+----------------|
| 1  | 1                 | CalciumHydroxide | -2.0      | 1.0            |
+----+-------------------+------------------+-----------+----------------+
```

special_reactant_compositions

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The special reactant composition identifier	
	✓		special_reactant_id	INTEGER	The id of the special reactant	
			element	VARCHAR	The name of the element in the special reactant	
			count	INTEGER	The stoichiometric count of the element in the special reactant	

Each special reactant has a fixed stoichiometric composition. Each `element` and its occurrence (`count`) in the reactant is stored in a row of the `special_reactant_composition` table.

Example:

```
SELECT c.*
 FROM special_reactant_compositions AS c
 JOIN special_reactants AS sr ON sr.id = c.special_reactant_id AND sr.variable_space_id = 1
 ORDER BY c.element;

+----+----------------------+---------+-------+
| id | special_reactant_id | element | count |
|----+----------------------+---------+-------|
| 1  | 1                    | Ca      | 1     |
| 3  | 1                    | H       | 2     |
| 2  | 1                    | O       | 2     |
+----+----------------------+---------+-------+
```

suppressions

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The suprression identifier	
	✓		variable_space_id	INTEGER	The id of the variable space point	
		✓	name	VARCHAR	The name of the species/phase to suppress, e.g. H_2O_2, methane, olivine	
		✓	type	VARCHAR	The type of species/phase to suppress, e.g. “solid solutions”	

- `CONSTRAINT suppressions_well_defined CHECK (name is not null or type is not null)`

Users can specify, within an order, that some species or entire phase type should be suppressed from forming. When a phase type is specified, exceptions can be provided. Those exceptions are stored separately in the suppression_exceptions table.

Example:

```
SELECT * FROM suppressions WHERE variable_space_id = 1;

+----+-------------------+------------+---------+
| id | variable_space_id | name       | type    |
|----+-------------------+------------+---------|
| 1  | 1                 | METHANE    | <null>  |
| 2  | 1                 | antigorite | <null>  |
| 3  | 1                 | <null>     | mineral |
+----+-------------------+------------+---------+
```

suppression_exceptions

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The suppression exception identifier	
			name	VARCHAR	The name the species/phase to except, e.g. H2O2, methane, olivine	
	✓		suppression_id	INTEGER	The id of the suppression for which to except	

Each exception to a suppression constraint is stored as a row in the `suppression_exceptions` table, referring back to the suppression by the `suppression_id` value.

Example:

```
SELECT se.*
 FROM suppression_exceptions AS se
 JOIN suppressions AS s ON s.id = se.suppression_id AND s.variable_space_id = 1
 ORDER BY se.name;

+----+-----------+----------------+
| id | name      | suppression_id |
|----+-----------+----------------|
| 1  | magnetite | 3              |
+----+-----------+----------------+
```

equilibrium_space

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium space point identifier	
	✓	✓	variable_space_id	INTEGER	The id of the variable space point for which the equilibrium was found	
			stage	VARCHAR	The stage of the equilibration, e.g. "eq3", "eq6"	
		✓	log_xi	DOUBLE	The logarithm of the ξ-step at which this equilibrium was found	unitless
			temperature	DOUBLE	The temperature	$^{\circ}C$
			pressure	DOUBLE	The pressure	bar
			"pH"	DOUBLE	The pH on the NBS scale	unitless
			"log_fO2"	DOUBLE	The logarithm of the oxygen (O_2) fugacity	$\log bar$
			log_activity_water	DOUBLE	The logarithm of the activity of water	unitless
			mole_fraction_water	DOUBLE	The mole-fraction of the fluid that is water	unitless
			log_gamma_water	DOUBLE	The logarithm of the activity coefficient of water	unitless
			"Eh"	DOUBLE	The redox potential	V
			pe	DOUBLE	The negative logarithm of the hypothetical electron activity	unitless
			"Ah"	DOUBLE	The redox affinity	$kcal$
		✓	"pcH"	DOUBLE	The pH based on the molarity of the hydrogen ion (H+)	unitless
		✓	"pHCl"	DOUBLE	The pHCl = pH + pCl, the sum of the negative logarithms of the activity of H+ and Cl- respectively	unitless
			log_ionic_strength	DOUBLE	The logarithm of the ionic strength of the solution	$\log molal$
			log_stoichiometric_ionic_strength	DOUBLE	The logarithm of the stoichiometric ionic strength of the solution	$\log molal$
			log_ionic_asymmetry	DOUBLE	The logarithm of the ionic asymmetry	$\log molal$
			log_stoichiometric_ionic_asymmetry	DOUBLE	The logarithm of the stoichiometric ionic asymmetry	$\log molal$
			osmotic_coefficient	DOUBLE	The osmotic coefficient of the solution	unitless
			stoichiometric_osmotic_coefficient	DOUBLE	The stoichiometric osmotic coefficient of the solution	unitless
			log_sum_molalities	DOUBLE	The logarithm of the sum of the molalities of all non-water species in the solution	$molal$
			log_sum_stoichiometric_molalities	DOUBLE	The logarithm of the sum of the stoichiometric molalities of all non-water species in the solution	$molal$
			charge_imbalance	DOUBLE	The charge imbalance of the solution	eq

PK	FK	N	Column Name	Type	Description	Unit
		✓	expected_charge_imbalance	DOUBLE	The charge imbalance of the solution before mass transfer	eq
		✓	sigma	DOUBLE	The total unsigned charge of the solution	eq
		✓	charge_discrepancy	DOUBLE	The charge discrepancy of the solution	eq
		✓	anions	DOUBLE	The total unsigned charge of anions per kilogram of water (σ_-)	eq/kg
		✓	cations	DOUBLE	the total unsigned charge of cations per kilogram of water (σ_+)	eq/kg
		✓	total_charge	DOUBLE	The total unsigned charge of the solution per kilogram of water ($\sigma = \sigma_+ + \sigma_-$)	eq/kg
		✓	mean_charge	DOUBLE	The mean charge of the solution per kilogram of water (0.5σ)	eq/kg
			solute_mass	DOUBLE	The total mass of all dissolved species	g
			solvent_mass	DOUBLE	The mass of the solvent (water)	g
			solution_mass	DOUBLE	The mass of the solution	g
		✓	solution_volume	DOUBLE	The volume of the solution	L
			tds	DOUBLE	The total dissolved solute (TDS); dissolved solute mass per kilogram of solution	mg/kg
			solute_fraction	DOUBLE	The fraction of the solution mass from dissolved species	unitless
			solvent_fraction	DOUBLE	The fraction of the solution mass from the solvent (water)	unitless
		✓	extended_alkalinity	DOUBLE	The extended alkalinity in units of electric charge per kilogram of H_2O	eq/kg
		✓	overall_affinity	DOUBLE	The sum of the molar affinities of all reactants	$kcal/mol$
		✓	reactant_mass_reacted	DOUBLE	The total mass reacted into the system up to this ξ step	g
		✓	reactant_mass_remaining	DOUBLE	The total mass remaining to be reacted	g
		✓	solid_mass_change	DOUBLE	The change of the solid mass of the system up to this ξ step	g
		✓	solid_mass_created	DOUBLE	The solid mass created up to this ξ step	g
		✓	solid_mass_destroyed	DOUBLE	The solid mass destroyed up to this ξ step	g
		✓	solid_volume_change	DOUBLE	The change of the solid volume of the system up to this ξ step	cm^3
		✓	solid_volume_created	DOUBLE	The solid volume created up to this ξ step	cm^3
		✓	solid_volume_destroyed	DOUBLE	The solid volume destroyed up to this ξ step	cm^3
			start_date	DATETIME	The datetime when the equilibration started	
			complete_date	DATETIME	The datetime when the equilibration completed	
			custom_properties	JSON	A JSON-encoded field for non-standard equilibrium properties computed by the kernel	

When **Eleanor** speciates a fluid, it is the kernel's responsibility to perform the calculations and provide one or more equilibrium space points. Each of these points, consisting of identifying information (`id`, `variable_space_id`, `stage` and `log_xi`) and a wide range of system properties (e.g. `temperature`, `pressure`, `"pH"`, etc...), is stored as a row in the `equilibrium_space` table. Additional, information about the system, such as the quantity of reactants reacted and remaining (`equilibrium_reactants`), the elemental composition of the fluid (`equilibrium_elements`), the molecular composition of the flud (`equilibrium_aqueous_species`), etc..., are stored in auxiliary tables that link back to the equilibrium space point via a foreign key reference on their `equilibrium_space_id` column.

For example, the builtin `eleanor.kernel.eq36.Kernel` will produce at least 2 equilibrium space points: one representing the speciation of the fluid alone (the `"eq3"` stage), and the final equilibrium after reacting the fluid with any reactants (the `"eq6"` stage). In the case that the user configured the kernel with the `track_paths: true` option, (many) more than 2 equilibrium space points will be produced: the `"eq3"` point, and some number of intermediate points between the `"eq3"` system and the final `"eq6"` representing a titration path. That is, each point along the titration path represents the equilibration of the fluid with an increasing quantity of reactant. The progress along the path is tracked by ξ (stored as `log_xi`). The `log_xi` value will be `NULL` for `"eq3"` stage points as ξ is undefined.

Different kernels, and different stages processed by a given kernel, may result in some columns being either `NULL` or `-inf` for one reason or another. This can cause confusion. Take for example the case of `log_xi` before. The `"eq3"` stage points will have `log_xi = NULL`, and the first point of a titration path produced in the `"eq6"` stage will have `log_xi = -inf` ($\xi = 0$ at the start of the path). However, if the user did not enable path tracking or set the kernel's print settings judiciously, then the first `"eq6"` stage point may have a finite `log_xi`. Other examples include:

1. Since mass is not transfered in the `"eq3"` stage, columns such as `solid_mass_change` will be `NULL`, and
2. Though they represent the the same quantity, more or less, the `"eq3"` stage outputs `sigma` while `"eq6"` outputs `total_charge` and only one of the two will be non-`NULL`.

Finally, each kernel may be implemented to compute and store non-standard properties, i.e. properties for which there is no `equilibrium_space` column or associated auxiliary table. These additional properties can be stored in JSON-encoded format in the `custom_properties` column.

Example (Without Paths):

```
SELECT
   id,
   variable_space_id,
   stage,
   log_xi,
   temperature,
   pressure,
   "pH",
   start_date,
   complete_date
 FROM equilibrium_space
 WHERE variable_space_id = 1
 ORDER BY stage, log_xi;
```

```
+----+-------------------+-------+---------------------+-------------+----------+--------+----------------------------+----------------------------+
| id | variable_space_id | stage | log_xi              | temperature | pressure | pH     | start_date                 | complete_date              |
|----+-------------------+-------+---------------------+-------------+----------+--------+----------------------------+----------------------------|
| 1  | 1                 | eq3   | <null>              | 300.0       | 500.0    | 5.4541 | 2026-01-09 10:59:02.54 | 2026-01-09 10:59:02.55 |
| 2  | 1                 | eq6   | -0.3000002024454176 | 300.0       | 500.0    | 5.3627 | 2026-01-09 10:59:02.55 | 2026-01-09 10:59:02.80 |
+----+-------------------+-------+---------------------+-------------+----------+--------+----------------------------+----------------------------+
```

Example (With Paths):

```
SELECT
   id,
   variable_space_id,
   stage,
   log_xi,
   temperature,
   pressure,
   "pH",
   start_date,
   complete_date
 FROM equilibrium_space
 WHERE variable_space_id = 1
 ORDER BY stage, log_xi;
```

```
+----+-------------------+-------+---------------------+-------------+----------+--------+-------------------------+-------------------------+
| id | variable_space_id | stage | log_xi              | temperature | pressure | pH     | start_date              | complete_date           |
|----+-------------------+-------+---------------------+-------------+----------+--------+-------------------------+-------------------------|
| 1  | 1                 | eq3   | <null>              | 300.0       | 500.0    | 5.4541 | 2026-01-09 11:02:59.49 | 2026-01-09 11:02:59.50 |
| 2  | 1                 | eq6   | -inf                | 300.0       | 500.0    | 5.4536 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 3  | 1                 | eq6   | -8.0                | 300.0       | 500.0    | 5.454  | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 4  | 1                 | eq6   | -7.0                | 300.0       | 500.0    | 5.4574 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 5  | 1                 | eq6   | -6.443549140627881  | 300.0       | 500.0    | 5.4706 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 6  | 1                 | eq6   | -6.0                | 300.0       | 500.0    | 5.5368 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 7  | 1                 | eq6   | -5.0                | 300.0       | 500.0    | 6.0213 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 8  | 1                 | eq6   | -4.734955647654595  | 300.0       | 500.0    | 6.1948 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 9  | 1                 | eq6   | -4.0                | 300.0       | 500.0    | 6.5286 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 10 | 1                 | eq6   | -3.8915466083537007 | 300.0       | 500.0    | 6.6473 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 11 | 1                 | eq6   | -3.22983825514633   | 300.0       | 500.0    | 6.3563 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 12 | 1                 | eq6   | -3.0                | 300.0       | 500.0    | 6.3854 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 13 | 1                 | eq6   | -2.0                | 300.0       | 500.0    | 6.2304 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 14 | 1                 | eq6   | -2.0                | 300.0       | 500.0    | 6.2304 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 15 | 1                 | eq6   | -1.8955811793524053 | 300.0       | 500.0    | 5.6519 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 16 | 1                 | eq6   | -1.4778975981455702 | 300.0       | 500.0    | 5.7453 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 17 | 1                 | eq6   | -1.4727156101843601 | 300.0       | 500.0    | 5.6938 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 18 | 1                 | eq6   | -1.3782070009986855 | 300.0       | 500.0    | 5.3294 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 19 | 1                 | eq6   | -1.0                | 300.0       | 500.0    | 5.3303 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
| 20 | 1                 | eq6   | -0.3000002024454176 | 300.0       | 500.0    | 5.3627 | 2026-01-09 11:02:59.50 | 2026-01-09 11:02:59.80 |
+----+-------------------+-------+---------------------+-------------+----------+--------+-------------------------+-------------------------+
```

equilibrium_reactants

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium reactant identifier	
	✓		equilibrium_space_id	INTEGER	The id of the equilibrium space point	
			name	VARCHAR	The name of the reactant	
			affinity	DOUBLE	The affinity of the reactant	
			relative_rate	DOUBLE	The molar rate of reactant titration relative to the total molar quantity of all reactants	mol/mol
			log_moles_reacted	DOUBLE	The moles of reactant that have been reacted up to this ξ step	$\log mol$
			log_moles_remaining	DOUBLE	The moles of reactant that remain to be reacted	$\log mol$
			log_mass_reacted	DOUBLE	The mass of the reactant that have been reacted up to this ξ step	$\log kg$
			log_mass_remaining	DOUBLE	The mass of the reactant that remain to be reacted	$\log kg$

If an equilibrium_space entry represents a stage in which mass is transfered, (i.e. reacted or precipitated) and the system has N reactants, then the point will be associated with N rows in the equilibrium_reactants table, one for each reactant.

> **Note**: The exception to this rule is for fixed_gas_reactants, which the eleanor.kernel.eq36 kernel treats differently. Fixed gas reactants are enforced by requiring equilibrium with a fictitious solid phase in equilibrium_pure_solids, e.g. CO2(g) as a fixed gas reactant will produce a pure solid of "fix_fCO2(g)", and do not appear in the equilibrium_reactants table.

The equilibrium_reactants table is most useful for two purposes:

1. **Quality assurance.** At the end of the equilibration, all the log_moles_remaining ought to be -inf (no reactant). If that is not the case, then the reaction did not progress far enough and kernel settings likely need to be tweaked.
2. **Path Tracking.** If titration path tracking is enabled, the log_moles_reacted should monotonically increase while log_moles_remaining monotonically decreases along the path (ordered by log_xi from the equilibrium_space table).

Example:

```
SELECT
   id,
   equilibrium_space_id AS eid,
   name,
   affinity,
   relative_rate AS rate,
   log_moles_reacted AS moles_reacted,
   log_moles_remaining,
   log_mass_reacted,
   log_mass_remaining
 FROM equilibrium_reactants
 WHERE equilibrium_space_id = 2
 ORDER BY name;
```

```
+----+-----+------------------+-----------+------+---------------------+---------------------+----------------------+--------------------+
| id | eid | name             | affinity  | rate | moles_reacted       | log_moles_remaining | log_mass_reacted     | log_mass_remaining |
|----+-----+------------------+-----------+------+---------------------+---------------------+----------------------+--------------------|
| 5  | 2   | CaCl2            | 9999999.0 | 0.0  | -2.0                | -inf                | 0.045283851395135605 | -inf               |
| 3  | 2   | CalciumHydroxide | 9999999.0 | 0.0  | -2.0                | -inf                | -0.13021109763031236 | -inf               |
| 1  | 2   | hematite         | 1.43      | 1.0  | -0.29999760285774607 | -inf                | 1.903285375549671    | -inf               |
| 6  | 2   | N2(g)            | 9999999.0 | 1.0  | -0.29999760285774607 | -inf                | 1.1473671077937864   | -inf               |
| 2  | 2   | olivine-ss       | 1.7744    | 1.0  | -0.29999760285774607 | -inf                | 1.9517453891382763   | -inf               |
| 4  | 2   | Si               | 9999999.0 | 0.0  | -2.0                | -inf                | -0.5515101084889142  | -inf               |
+----+-----+------------------+-----------+------+---------------------+---------------------+----------------------+--------------------+
```

equilibrium_elements

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The equilibrium element identifier	
	✓		`equilibrium_space_id`	`INTEGER`	The id of the equilibrium space point	
			`name`	`VARCHAR`	The name of the element (atomic symbol)	
			`log_molality`	`DOUBLE`	The molality of the element in solution	$\log molal$
			`mass_fraction`	`DOUBLE`	The fraction of solution mass accounted for by ths element	unitless

Every `equilibrium_space` entry will be associated with some number of rows in the `equilibrium_elements` table, one for each element in the fluid. These rows record the `name` and `log_molality` of the element in the fluid and the `mass_fraction` of the fluid that element represents.

If no precipitates form, then these rows should agree with the entries in the `elements` table. However, that is rarely the case, so expect them to differ.

Example:

```
SELECT *
 FROM equilibrium_elements
 WHERE equilibrium_space_id = 2
 ORDER BY name;

+----+----------------------+------+----------------------+------------------------+
| id | equilibrium_space_id | name | log_molality         | mass_fraction          |
|----+----------------------+------+----------------------+------------------------|
| 16 | 2                    | C    | -5.5935330597128265  | 3.01624e-08            |
| 14 | 2                    | Ca   | -2.1571530866467845  | 0.000274913            |
| 12 | 2                    | Cl   | -1.6971649242940714  | 0.000701302            |
| 15 | 2                    | Fe   | -6.133895871377041   | 4.04134e-08            |
| 13 | 2                    | H    | 2.0457140589408676   | 0.110299               |
| 17 | 2                    | Mg   | -4.097879571162284   | 1.9108999999999998e-06 |
| 19 | 2                    | N    | 0.002835353178722796 | 0.0138865              |
| 18 | 2                    | Na   | -8.998192757497696   | 2.2738399999999998e-11 |
| 11 | 2                    | O    | 1.7443829627474854   | 0.8747969999999999     |
| 20 | 2                    | Si   | -2.838188943951825   | 4.01526e-05            |
+----+----------------------+------+----------------------+------------------------+
```

equilibrium_aqueous_species

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium aqueous species identifier	
	✓		equilibrium_space_id	INTEGER	The id of the equilibrium space point	
			name	VARCHAR	The name of the aqueous species, e.g. “H2O2”	
			log_molality	DOUBLE	The logarithm of the molality of that species in solution	$\log molal$
			log_activity	DOUBLE	The logarithm of the activity of that species in solution	unitless
			log_gamma	DOUBLE	The logarithm of the activity coefficient of that species in solution	unitless

Every equilibrium_space entry will be associated with some number of rows in the equilibrium_aqueous_species table, one for each species in the fluid. These rows record the name, log_molality, log_activity and log_gamma (log-activity coefficient) of the species in accordance with the activity model used by the kernel.

Example:

```
SELECT *
 FROM equilibrium_aqueous_species
 WHERE equilibrium_space_id = 2
 ORDER BY log_molality DESC, name
 LIMIT 10;

+----+----------------------+-------+--------------+--------------+-----------+
| id | equilibrium_space_id | name  | log_molality | log_activity | log_gamma |
|----+----------------------+-------+--------------+--------------+-----------|
| 65 | 2                    | N2    | -0.3092      | -0.3092      | 0.0       |
| 66 | 2                    | NH3   | -1.7195      | -1.7195      | 0.0       |
| 67 | 2                    | Cl-   | -1.7863      | -1.9078      | -0.1215   |
| 68 | 2                    | NH4+  | -2.2149      | -2.3465      | -0.1316   |
| 69 | 2                    | CaCl+ | -2.4406      | -2.5666      | -0.126    |
| 70 | 2                    | Ca+2  | -2.4947      | -2.9493      | -0.4545   |
| 71 | 2                    | SiO2  | -2.8385      | -2.8385      | 0.0       |
| 72 | 2                    | H2    | -3.1232      | -3.1179      | 0.0052    |
| 73 | 2                    | CaOH+ | -4.0071      | -4.1331      | -0.126    |
| 74 | 2                    | Mg+2  | -4.3002      | -4.7213      | -0.4211   |
+----+----------------------+-------+--------------+--------------+-----------+
```

equilibrium_gases

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The equilibrium gas identifier	
	✓		`equilibrium_space_id`	`INTEGER`	The id of the equilibrium space point	
			`name`	`VARCHAR`	The name of the gas, e.g. “H2(g)”	
			`log_fugacity`	`DOUBLE`	The logarithm of the fugacity of the gas at this ξ step	$\log bar$

Every `equilibrium_space` entry will be associated with some number of rows in the `equilibrium_gases` table, one for each gas phase. These rows record the `name` and `log_fugacity`.

Example:

```
SELECT *
 FROM equilibrium_gases
 WHERE equilibrium_space_id = 2
 ORDER BY log_fugacity DESC, name;

+----+----------------------+--------+--------------+
| id | equilibrium_space_id | name   | log_fugacity |
|----+----------------------+--------+--------------|
| 1  | 2                    | N2(g)  | 2.41361      |
| 2  | 2                    | H2O(g) | 1.85299      |
| 3  | 2                    | H2(g)  | -0.58253     |
| 4  | 2                    | NH3(g) | -0.83984     |
| 5  | 2                    | CO2(g) | -3.5         |
| 6  | 2                    | CH4(g) | -4.06719     |
| 7  | 2                    | CO(g)  | -7.60332     |
| 8  | 2                    | NO(g)  | -23.50603    |
| 9  | 2                    | N2O(g) | -26.10458    |
| 10 | 2                    | O2(g)  | -34.26937    |
+----+----------------------+--------+--------------+
```

equilibrium_pure_solids

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium pure solid id	
	✓		equilibrium_space_id	INTEGER	The id of the equilibrium space point	
			name	VARCHAR	The name of the pure solid, e.g. “fayalite”	
			log_qk	DOUBLE	The saturation index of the pure solid phase	unitless
			affinity	DOUBLE	the affinity of the pure solid	$kcal$
			log_moles	DOUBLE	The logarithm of the net moles of the solid that have precipitated up to this ξ step	$\log mol$
			log_mass	DOUBLE	The logarithm of the net mass of the solid that has precipitated up to this ξ step	$\log g$
			log_volume	DOUBLE	The logarithm of the volume of the solid that has precipitated	$\log cm^3$

Every equilibrium_space entry will be associated with some number of rows in the equilibrium_pure_solids table, one for each pure solid (not a solid solution). These rows record the name, saturation index (log_qk), affinity, log_moles, log_mass and log_volume of the solid phase.

Notes:

1. The solids that appear in this table include both solids that form as well as those that *could* have formed based on the composition of the system. For example, the solids associated with the `"eq3"` stage will all be hypothetical as the kernel does not partition mass in that stage. As such, the value in the `log_moles`, `log_mass` and `log_volume` columns will be `-inf`. Similarly, those columns will be `-inf` for `"eq6"`-stage equilibrium space points if the phase does not actually precipitate.
2. In many cases, volume data is lacking. Even if a solid precipitates, it will *likely* have a $V = 0$ (`log_volume = -inf`).

Example:

```
SELECT *
 FROM equilibrium_pure_solids
 WHERE equilibrium_space_id = 2
 ORDER BY log_mass DESC, log_qk DESC, name
 LIMIT 10;
```

```
+-----+----------------------+------------+----------+----------+-----------+--------------------+------------+
| id  | equilibrium_space_id | name       | log_qk   | affinity | log_moles | log_mass           | log_volume |
|-----+----------------------+------------+----------+----------+-----------+--------------------+------------|
| 56  | 2                    | magnetite  | -0.0     | -0.0     | -0.2891   | 2.0755104645244136 | -inf       |
| 62  | 2                    | antigorite | 1.67537  | 4.39389  | -inf      | -inf               | -inf       |
| 110 | 2                    | tremolite  | -0.0     | -0.0     | -inf      | -inf               | -inf       |
| 109 | 2                    | talc       | -0.12292 | -0.32237 | -inf      | -inf               | -inf       |
| 68  | 2                    | chrysotile | -0.30052 | -0.78816 | -inf      | -inf               | -inf       |
| 74  | 2                    | diopside   | -0.42799 | -1.12247 | -inf      | -inf               | -inf       |
| 57  | 2                    | hematite   | -0.54526 | -1.43002 | -inf      | -inf               | -inf       |
| 78  | 2                    | enstatite  | -0.739   | -1.93813 | -inf      | -inf               | -inf       |
| 105 | 2                    | quartz     | -0.8933  | -2.34282 | -inf      | -inf               | -inf       |
| 88  | 2                    | goethite   | -0.90359 | -2.36978 | -inf      | -inf               | -inf       |
+-----+----------------------+------------+----------+----------+-----------+--------------------+------------+
```

equilibrium_solid_solutions

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The equilibrium solid solution identifier	
	✓		`equilibrium_space_id`	`INTEGER`	The id of the equilibrium space point	
			`name`	`VARCHAR`	The name of the solid solution, e.g. “olivine”	
			`log_qk`	`DOUBLE`	The saturation index of the solid solution	unitless
			`affinity`	`DOUBLE`	The affinity of the solid solution	$kcal$
			`log_moles`	`DOUBLE`	The logarithm of the net moles of the solid solution that have precipitated up to this ξ step	$\log mol$
			`log_mass`	`DOUBLE`	The logarithm of the net mass of the solid solution that has precipitated up to this ξ step	$\log g$
			`log_volume`	`DOUBLE`	The logarithm of the volume of the solid solution that has precipitated	$\log cm^3$

Every equilibrium_space entry will be associated with some number of rows in the equilibrium_solid_solutions table, one for each solid solution. These rows record the name, saturation index (log_qk), affinity, log_moles, log_mass and log_volume of the solid solution. Additionally, the end member composition of the solid solutions are stored in the equilibrium_end_members auxiliary table.

Notes:

1. The solid solutions that appear in this table include both solid solutions that form as well as those that *could* have formed based on the composition of the system. For example, the solid solutions associated with the "eq3" stage will all be hypothetical as the kernel does not partition mass in that stage. As such, the value in the log_moles, log_mass and log_volume columns will be -inf. Similarly, those columns will be -inf for "eq6"-stage equilibrium space points if the phase does not actually precipitate.
2. In many cases, volume data is lacking. Even if a solid precipitates, it will *likely* have a $V = 0$ (log_volume = -inf).

Example:

```
SELECT *
 FROM equilibrium_solid_solutions
 WHERE equilibrium_space_id = 2
 ORDER BY log_mass DESC, log_qk DESC, name
 LIMIT 10;
```

id	equilibrium_space_id	name	log_qk	affinity	log_moles	log_mass	log_volume
13	2	talc-ss	-0.0	-0.0	-1.0435	1.5450472044011843	-inf
12	2	serpentine	-0.0	-0.0	-1.2932	1.1838390370564211	-inf
11	2	amphib_ternary-ss	-0.0	-0.0	-2.1849	0.7248327204665614	-inf
19	2	ca-amphib_binary-ss	-0.0	-0.0	-inf	-inf	-inf
18	2	clinopyroxene-ss	-0.41699	-1.09361	-inf	-inf	-inf
17	2	orthopyroxene-ss	-0.59034	-1.54825	-inf	-inf	-inf
16	2	ideal_olivine-ss	-0.64149	-1.6824	-inf	-inf	-inf
14	2	olivine-ss	-0.64149	-1.6824	-99999.0	-inf	-inf
20	2	brucite-ss	-1.2999	-3.40916	-inf	-inf	-inf
15	2	carbonate-ss	-4.07608	-10.69009	-inf	-inf	-inf

equilibrium_end_members

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium solid solution end member identifier	
	✓		equilibrium_solid_solution_id	INTEGER	The id of the equilibrium solid solution	
			name	VARCHAR	The name of the end member, e.g. “fayalite”	
			log_qk	DOUBLE	The saturation index of the end member	unitless
			affinity	DOUBLE	The affinity of the end member	$kcal$
			log_moles	DOUBLE	The logarithm of the net moles of the end member that have precipitated up to this ξ step	$\log mol$
			log_mass	DOUBLE	The logarithm of the net mass of the end member that has precipitated up to this ξ step	$\log g$
			log_volume	DOUBLE	The logarithm of the volume of the end member that has precipitated	$\log cm^3$

Every equilibrium_solid_solutions entry will be associated with some number of rows in the equilibrium_pure_solids table, one for each solid solution. These rows record the name, saturation index (log_qk), affinity, log_moles, log_mass and log_volume of the solid phase.

Notes:

1. The end members that appear in this table include both end members that form as well as those that *could* have formed based on the composition of the system. For example, end members of solid solutions associated with the "eq3" stage will all be hypothetical as the kernel does not partition mass in that stage. As such, the value in the log_moles, log_mass and log_volume columns will be -inf. Similarly, those columns will be -inf for "eq6"-stage equilibrium space points if the phase does not actually precipitate.
2. In many cases, volume data is lacking. Even if a solid precipitates, it will *likely* have a $V = 0$ (log_volume = -inf).

Example:

```
SELECT *
 FROM equilibrium_end_members
 WHERE equilibrium_solid_solution_id = 13
 ORDER BY log_mass DESC, log_qk DESC, name;

+----+-------------------------------+-------------+--------+----------+-----------+--------------------+------------+
| id | equilibrium_solid_solution_id | name        | log_qk | affinity | log_moles | log_mass           | log_volume |
|----+-------------------------------+-------------+--------+----------+-----------+--------------------+------------|
| 28 | 13                            | talc        | -0.0   | -0.0     | -1.0845   | 1.4944328987263986 | -inf       |
| 29 | 13                            | minnesotaite | -0.0   | -0.0     | -2.0892   | 0.5865197925102743 | -inf       |
+----+-------------------------------+-------------+--------+----------+-----------+--------------------+------------+
```

equilibrium_redox_reactions

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium redox reaction identifier	
	✓		equilibrium_space_id	INTEGER	The id of the equilibrium space point	
			couple	VARCHAR	The name of the redox couple	
			"Eh"	DOUBLE	The redox potential	V
			pe	DOUBLE	The negative logarithm of the hypothetical electron activity	unitless
			"log_fO2"	DOUBLE	The logarithm of the oxygen (O_2) fugacity	$\log bar$
			"Ah"	DOUBLE	The redox affinity	$kcal$

Each equilibrium_space entry can be associated with one or more rows in the equilibrium_redox_reactions table, one for each redox constraint on the system.

> **Note**: In the present version of **Eleanor**, there will typically a single equilibrium_redox_reactions entry for each equilibrium_space entry, and the values in this table will be wholly redundant with equilibrium_space.

Example:

```
SELECT redox.*
 FROM equilibrium_redox_reactions AS redox
 JOIN equilibrium_space AS es ON es.id = redox.equilibrium_space_id AND es.variable_space_id = 1;

+----+----------------------+---------+--------+---------+---------+---------+
| id | equilibrium_space_id | couple  | Eh     | pe      | log_fO2 | Ah      |
|----+----------------------+---------+--------+---------+---------+---------|
| 1  | 1                    | DEFAULT | 0.373  | 3.2836  | -1.0    | 8.612   |
| 2  | 2                    | DEFAULT | -0.562 | -4.9403 | -34.269 | -12.957 |
+----+----------------------+---------+--------+---------+---------+---------+
```

Appendix - Example Order

```yaml
name: A Wild Example Order
creator: 39 Alpha Research, Team One
date: 2026-01-09
notes: This is a nonsensical order intended to demonstrate a wide

kernel:
  type: eq36
  model: b-dot
  charge_balance: H+
  track_paths: true
  eq6_config:
    xi_max: 1
    print_step_interval: 1

elements:
  C: -5
  Mg: -9
  Fe: -9
  Na: [-4, -5]
  Si:
    min: -9
    max: -6
  Ca:
    mean: -4
    stddev: 0.01
    min: -5
    max: -3

species:
  H+: -7
  O2(g): -1
reactants:
  CaCl2:
    type: aqueous
    amount: -3.0
    titration_rate: 1.0
  Na:
    type: element
    amount: -4.0
    titration_rate: 1.0
  CO2(g):
    type: fixed gas
    amount: -0.3
    log_fugacity: -3.5
  N2(g):
    type: gas
    amount: -0.3
    titration_rate: 1.0
  hematite:
    type: mineral
    amount: -0.3
    titration_rate: 1.0
  olivine-ss:
    type: solid solution
    amount: -0.3
    titration_rate: 1.0
    end_members:
      fayalite: 0.6
      forsterite: 0.4
  CalciumHydroxide:
    type: special
    amount: -0.3
    titration_rate: 1.0
    composition:
      Ca: 1
      O: 2
      H: 2

suppressions:
  - METHANE
  - antigorite
  - type: mineral
    except:
      - magnetite
```