Table 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 form 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. `-Infinity`. Columns that are nullable (**N**) are denoted as such.

orders

PK	FK	N	Column Name	Type	Description	Unit
✓			`id`	`INTEGER`	The order identifier	

PK	FK	N	Column Name	Type	Description	Unit
			`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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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 variable space 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** on an order 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 and 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 alongside in the variable_space table as well.

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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.

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
```

```
| 2  | 1                | forsterite | 1.02329   | 1.0            |
+----+------------------+------------+-----------+----------------+
```

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. These are provided inside of the order file via the `kernel` block, e.g.

```
kernel:
  type: eq36
  model: b-dot
  charge_balance: H+
```

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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+------------------+------------+-----------+----------------|
| 1  | 1                | fayalite   | 1.02329   | 1.0            |
| 2  | 1                | forsterite | 1.02329   | 1.0            |
+----+------------------+------------+-----------+----------------+
```

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)	

PK	FK	N	Column Name	Type	Description	Unit
			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. Within the order specification, this is done in the `elements` block, e.g.

```
elements:
  C: -9
  Na: [-2, -3]
  Si:
    min: -9
    max: -6
  Ca:
    mean: -4
    stddev: 0.01
    min: -5
    max: -3
```

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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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 (formula)	
			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. Within the order specification, this is done in the `species` block, e.g.

```
species:
  H+: -7     # pH ~ -log activity H+
  O2(g): -1  # log fO2
```

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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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	

PK	FK	N	Column Name	Type	Description	Unit
	✓		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`).

Within the order specification, this is done in the `reactants` block, e.g.

```
reactants:
  CaCl2:
    type: aqueous
    amount: -3.0        # log moles of CaCl2 titrated into the system
    titration_rate: 1.0 # moles of CaCl2 titrated into the system by ξ=1
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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`).

Within the order specification, this is done in the `reactants` block, e.g.

```
reactants:
  Na:
    type: element
    amount: -4.0         # log moles of Na added into the system
    titration_rate: 1.0  # moles of Na added into the system by ξ=1
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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 reactant to be reacted with the fluid	$\log mol$
			`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`.

Within the order specification, this is done in the `reactants` block, e.g.

```
reactants:
  CO2(g):
    type: fixed gas
    amount: -0.3  # log moles of CO2(g) in the buffer
    log_fugacity: -3.5
  O2(g):
    type: fixed gas
    amount: -0.3  # log moles of O2(g) in the buffer
    log_fugacity: -0.7
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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 gas with the fluid at some ξ-rate (`titration_rate`).

Within the order specification, this is done in the `reactants` block, e.g.

```
reactants:
  N2(g):
    type: gas
    amount: -0.3
    titration_rate: 1.0
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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`).

Within the order specification, this is done in the `reactants` block, e.g.

```yaml
reactants:
  fayalite:
    type: mineral
    amount: -0.3
    titration_rate: 1.0
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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 molar fraction of each of its end members. The end member data is stored separately in the `solid_solution_reactant_end_members` table.

Within the order specification, this is done in the `reactants` block, e.g.

```
reactants:
  olivine-ss:
    type: solid solution
    amount: -0.3
    titration_rate: 1.0
    end_members:
      fayalite: 0.6
      forsterite: 0.4
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
```

```
| 2  | 1                | forsterite | 1.02329   | 1.0            |
+----+------------------+------------+-----------+----------------+
```

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.

Within the order specification, this is done in the `reactants` block, e.g.

```yaml
reactants:
  CalciumHydroxide:
    type: special
    amount: -0.3
    titration_rate: 1.0
    composition:
      Ca: 1
      O: 2
      H: 2
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+------------------+------------+-----------+----------------|
| 1  | 1                | fayalite   | 1.02329   | 1.0            |
| 2  | 1                | forsterite | 1.02329   | 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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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.

Within the order specification, this is done in the `suppressions` block, e.g.

```
suppressions:
  - METHANE               # suppress methane formation
  - antigorite            # suppress antigorite precipitation
  - type: solid solutions  # suppress precipitation of all solid solutions
  - type: mineral          # suppress all mineral precipitation *except* magnetite
```

```
    except:
      - magnetite
```

Example:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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:

```
example=# SELECT * FROM mineral_reactants WHERE variable_space_id = 1;
+----+-------------------+------------+-----------+----------------+
| id | variable_space_id | name       | log_moles | titration_rate |
|----+-------------------+------------+-----------+----------------|
| 1  | 1                 | fayalite   | 1.02329   | 1.0            |
| 2  | 1                 | forsterite | 1.02329   | 1.0            |
+----+-------------------+------------+-----------+----------------+
```

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

PK	FK	N	Column Name	Type	Description	Unit
			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
		✓	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

PK	FK	N	Column Name	Type	Description	Unit
		✓	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 ($0.5\sigma_\pm$)	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 equilibrium properties computed by the kernel	

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	

PK	FK	N	Column Name	Type	Description	Unit
			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	$\log kg$

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

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

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	

PK	FK	N	Column Name	Type	Description	Unit
			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	$\log bar$

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	$\log cm^3$

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	$\log cm^3$

equilibrium_end_members

PK	FK	N	Column Name	Type	Description	Unit
✓			id	INTEGER	The equilibrium solid solution end member identifier	

PK	FK	N	Column Name	Type	Description	Unit
	✓		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	$\log cm^3$

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	$kcal$