New in v2.0：JSONB数据类型将JSON（JavaScript Object Notation）数据作为JSONB值的二进制存储，它消除了空格，重复键和键排序。 JSONB支持倒排索引。

有关从第三方API存储和查询JSON数据的示例，请参阅[JSON教程](https://www.cockroachlabs.com/docs/v2.0/demo-json-support.html)。

别名

在CockroachDB, JSON是 JSONB的别名。

在PostgreSQL中，JSONB和JSON是两种不同的数据类型。 在CockroachDB中，JSONB / JSON数据类型的行为与PostgreSQL中的JSONB数据类型相似。

注意事项

* 主键，外键和唯一约束不能用于JSONB值。
* 无法在JSONB列上创建标准索引; 你必须使用倒排索引。

语法

JSONB数据类型的语法遵循RFC8259中指定的格式。JSONB类型的常量值可以使用解释文本（interpreted literal）或使用JSONB类型注释的字符串文本来表示。

有六种类型的JSONB值：

* null
* Boolean
* String
* Number (i.e., decimal，不是标准 int64)
* Array (即有序的JSONB值序列)
* Object (即从字符串到JSONB值的映射)

例如:

* '{"type": "account creation", "username": "harvestboy93"}'
* '{"first\_name": "Ernie", "status": "Looking for treats", "location" : "Brooklyn"}'

如果输入中包含重复键，则仅保留最后一个值。

大小

JSONB值的大小是可变的，但建议将值保持在1 MB以下以保证性能。 如果高于该阈值，写入放大和其他考虑因素可能导致明显的性能下降。

JSONB 函数

| **Function** | **Description** |
| --- | --- |
| jsonb\_array\_elements(<jsonb>) | 将JSONB数组扩展为一组JSONB值。 |
| jsonb\_build\_object(<any\_element>...) | 从可变参数列表构建一个JSONB对象，该列表在键和值之间交替。 |
| jsonb\_each(<jsonb>) | 将最外面的JSONB对象扩展为一组键值对。 |
| jsonb\_object\_keys(<jsonb>) | 返回最外面的JSONB对象中的有序键的集合。 |
| jsonb\_pretty(<jsonb>) | 返回给定的JSONB值作为缩进的STRING，并带有换行符。 请参阅下面的示例。 |

有关支持的JSONB函数的完整列表， 查看 [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/#jsonb-functions).

JSONB 操作符

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| -> | 访问JSONB字段，返回JSONB值。 | SELECT '{"foo":"bar"}'::JSONB->'foo' = '"bar"'::JSONB; |
| ->> | 访问JSONB字段，返回一个字符串。 | SELECT '{"foo":"bar"}'::JSONB->>'foo' = 'bar'::STRING; |
| @> | 测试左侧的JSONB字段是否包含右侧的的JSONB字段。 | SELECT ('{"foo": {"baz": 3}, "bar": 2}'::JSONB @> '{"foo": {"baz":3}}'::JSONB ) = true; |

有关支持的JSONB运算符的完整列表，查看 [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/).

示例

使用JSONB列创建表

> **CREATE** **TABLE** users (

profile\_id **UUID** **PRIMARY** **KEY** **DEFAULT** gen\_random\_uuid(),

last\_updated **TIMESTAMP** **DEFAULT** **now**(),

user\_profile JSONB

);

> **SHOW** **COLUMNS** **FROM** users;

+--------------+-----------+-------+-------------------+-------------+

| Field | Type | Null | Default | Indices |

+--------------+-----------+-------+-------------------+-------------+

| profile\_id | UUID | false | gen\_random\_uuid() | {"primary"} |

| last\_updated | TIMESTAMP | true | now() | {} |

| user\_profile | JSON | true | NULL | {} |

+--------------+-----------+-------+-------------------+-------------+

> **INSERT** **INTO** users (user\_profile) **VALUES**

('{"first\_name": "Lola", "last\_name": "Dog", "location": "NYC", "online" : true, "friends" : 547}'),

('{"first\_name": "Ernie", "status": "Looking for treats", "location" : "Brooklyn"}');

> **SELECT** \* **FROM** users;

+--------------------------------------+----------------------------------+--------------------------------------------------------------------------+

| profile\_id | last\_updated | user\_profile |

+--------------------------------------+----------------------------------+--------------------------------------------------------------------------+

| 33c0a5d8-b93a-4161-a294-6121ee1ade93 | 2018-02-27 16:39:28.155024+00:00 | {"first\_name": "Lola", "friends": 547, "last\_name": "Dog", "location": |

| | | "NYC", "online": true} |

| 6a7c15c9-462e-4551-9e93-f389cf63918a | 2018-02-27 16:39:28.155024+00:00 | {"first\_name": "Ernie", "location": "Brooklyn", "status": "Looking for |

| | | treats"} |

+--------------------------------------+----------------------------------+--------------------------------------------------------------------------+

检索格式化的JSONB数据

要使用更易于阅读的格式检索JSONB数据，请使用jsonb\_pretty()函数。 例如，从你在第一个示例中创建的表中检索数据：

> **SELECT** profile\_id, last\_updated, jsonb\_pretty(user\_profile) **FROM** users;

+--------------------------------------+----------------------------------+------------------------------------+

| profile\_id | last\_updated | jsonb\_pretty |

+--------------------------------------+----------------------------------+------------------------------------+

| 33c0a5d8-b93a-4161-a294-6121ee1ade93 | 2018-02-27 16:39:28.155024+00:00 | { |

| | | "first\_name": "Lola", |

| | | "friends": 547, |

| | | "last\_name": "Dog", |

| | | "location": "NYC", |

| | | "online": true |

| | | } |

| 6a7c15c9-462e-4551-9e93-f389cf63918a | 2018-02-27 16:39:28.155024+00:00 | { |

| | | "first\_name": "Ernie", |

| | | "location": "Brooklyn", |

| | | "status": "Looking for treats" |

| | | } |

+--------------------------------------+----------------------------------+------------------------------------+

从JSONB值中检索特定字段

要从JSONB值检索特定字段，请使用->运算符。 例如，从你在第一个示例中创建的表中检索字段：

> **SELECT** user\_profile->'first\_name',user\_profile->'location' **FROM** users;

+----------------------------+--------------------------+

| user\_profile->'first\_name' | user\_profile->'location' |

+----------------------------+--------------------------+

| "Lola" | "NYC" |

| "Ernie" | "Brooklyn" |

+----------------------------+--------------------------+

你还可以使用 ->> 运算符将 JSONB 字段值作为STRING值返回：

> **SELECT** user\_profile->>'first\_name', user\_profile->>'location' **FROM** users;

+-----------------------------+---------------------------+

| user\_profile->>'first\_name' | user\_profile->>'location' |

+-----------------------------+---------------------------+

| Lola | NYC |

| Ernie | Brooklyn |

+-----------------------------+---------------------------+

有关我们支持的函数和操作符的完整列表，查看 [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/).

使用JSONB列和计算列创建表

在这个例子中，让我们创建一个包含JSONB列和计算列的表：

**CREATE** **TABLE** student\_profiles (

id **STRING** **PRIMARY** **KEY** **AS** (**profile**->>'id') STORED,

**profile** JSONB

);

然后插入几行数据：

INSERT INTO student\_profiles (profile) VALUES

('{"id": "d78236", "name": "Arthur Read", "age": "16", "school": "PVPHS", "credits": 120, "sports": "none"}'),

('{"name": "Buster Bunny", "age": "15", "id": "f98112", "school": "THS", "credits": 67, "clubs": "MUN"}'),

('{"name": "Ernie Narayan", "school" : "Brooklyn Tech", "id": "t63512", "sports": "Track and Field", "clubs": "Chess"}');

**SELECT** \* **FROM** student\_profiles;

+--------+---------------------------------------------------------------------------------------------------------------------+

| id | profile |

+--------+---------------------------------------------------------------------------------------------------------------------+

| d78236 | {*"age"*: *"16"*, *"credits"*: 120, *"id"*: *"d78236"*, *"name"*: *"Arthur Read"*, *"school"*: *"PVPHS"*, *"sports"*: *"none"*} |

| f98112 | {*"age"*: *"15"*, *"clubs"*: *"MUN"*, *"credits"*: 67, *"id"*: *"f98112"*, *"name"*: *"Buster Bunny"*, *"school"*: *"THS"*} |

| t63512 | {*"clubs"*: *"Chess"*, *"id"*: *"t63512"*, *"name"*: *"Ernie Narayan"*, *"school"*: *"Brooklyn Tech"*, *"sports"*: *"Track and Field"*} |

+--------+---------------------------------------------------------------------------------------------------------------------+

主key id被作为profile列中的字段计算。

支持的转换

JSONB值可以转换为以下数据类型：

* STRING

See Also

* [JSON tutorial](http://doc.cockroachchina.baidu.com/#quick-start/explore-benefits/json-support/)
* [Inverted Indexes](http://doc.cockroachchina.baidu.com/#performance-optimization/inverted-indexes/)
* [Computed Columns](http://doc.cockroachchina.baidu.com/#develop/sql-statements/CONPUTED-COLUMNS/)
* [Data Types](http://doc.cockroachchina.baidu.com/#develop/data-types/overview/)
* [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/)