PGCON 2023

CIPHER DOC

A Searchable, Encrypted JSON Document Service on Postgres

David E. Wheeler
justatheory.com
david.wheeler@nytimes.com

TEAM PRINCIPLES

TEAM PRINCIPLES

- Put our users’ intentions first

TEAM PRINCIPLES

- Put our users' intentions first
- Foster continuity across experiences

TEAM PRINCIPLES

- Put our users' intentions first
- Foster continuity across experiences
- Empower users with transparency and control

TEAM PRINCIPLES

- Put our users' intentions first
- Foster continuity across experiences
- Empower users with transparency and control
- Protect data entrusted to us

TEAM PRINCIPLES

- Put our users’ intentions first
- Foster continuity across experiences
- Empower users with transparency and control
- Protect data entrusted to us
- Build together

CENTRALIZED DOCUMENT MANAGEMENT

CENTRALIZED DOCUMENT MANAGEMENT

- Standard, accessible Web API

CENTRALIZED DOCUMENT MANAGEMENT

- Standard, accessible Web API
- Schema validation/enforcement

CENTRALIZED DOCUMENT MANAGEMENT

- Standard, accessible Web API
- Schema validation/enforcement
- Worthless in breach

CENTRALIZED DOCUMENT MANAGEMENT

- Standard, accessible Web API
- Schema validation/enforcement
- Worthless in breach
- Secondary key search

{REST}

AN INTUITIVE CRUD INTERFACE

AN INTUITIVE CRUD INTERFACE

- Basic CRUD

AN INTUITIVE CRUD INTERFACE

- Basic CRUD
- Fetch by ID

AN INTUITIVE CRUD INTERFACE

- Basic CRUD
- Fetch by ID
- Simple key/value design

AN INTUITIVE CRUD INTERFACE

- Basic CRUD
- Fetch by ID
- Simple key/value design
- Boringly RESTful

SCHEMA

SCHEMA

```
CREATE TABLE users (
    id     UUID PRIMARY KEY,
    entity JSONB
);
```

SCHEMA

```
CREATE TABLE users (
    id      UUID PRIMARY KEY,
    entity JSONB
);
```

SCHEMA

```sql
CREATE TABLE users (
    id        UUID PRIMARY KEY,
    entity JSONB
);
```

REST DEMO

PROTECT DATA ENTRUSTED TO US

REQUIREMENT: DATABASE DUMP WORTHLESS TO BREACHERS

IMPLICATION: NO PLAIN TEXT DATA IN THE DATABASE

ENCRYPTION PATTERN

ENCRYPTION PATTERN

- Use AEAD encryption

ENCRYPTION PATTERN

- Use AEAD encryption
  - Industry standard

ENCRYPTION PATTERN

- Use AEAD encryption
  - Industry standard
  - Authenticated

ENCRYPTION PATTERN

- Use AEAD encryption
  - Industry standard
  - Authenticated
  - Additional data

ENCRYPTION PATTERN

- Use AEAD encryption
  - Industry standard
  - Authenticated
  - Additional data
- Entity fully encrypted

ENCRYPTION PATTERN

- Use AEAD encryption
  - Industry standard
  - Authenticated
  - Additional data
- Entity fully encrypted
- ID unencrypted

SCHEMA

```sql
CREATE TABLE users (
    id     UUID PRIMARY KEY,
    entity JSONB
);
```

SCHEMA

```
CREATE TABLE users (
    id        UUID PRIMARY KEY,
    entity JSONB
);
```

SCHEMA

```
CREATE TABLE users (
    id     UUID PRIMARY KEY,
    entity BYTEA
);
```

SO HOW DO WE DO THAT?

GOOGLE TINK

TINK

TINK

- Open-Source cryptography library

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers
- Secure & simple cryptographic APIs

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers
- Secure & simple cryptographic APIs
  - User-centered design

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers
- Secure & simple cryptographic APIs
  - User-centered design
  - Reduce common pitfalls

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers
- Secure & simple cryptographic APIs
  - User-centered design
  - Reduce common pitfalls
  - Careful implementation and code reviews

TINK

- Open-Source cryptography library
- By Google cryptographers and security engineers
- Secure & simple cryptographic APIs
  - User-centered design
  - Reduce common pitfalls
  - Careful implementation and code reviews
  - Extensive testing

TINK

TINK

- Safely implement common cryptographic tasks

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google
- AEAD, HMAC, key rotation

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google
- AEAD, HMAC, key rotation
- Requires keys encrypted by KMS

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google
- AEAD, HMAC, key rotation
- Requires keys encrypted by KMS
- Encapsulates ciphertext format

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google
- AEAD, HMAC, key rotation
- Requires keys encrypted by KMS
- Encapsulates ciphertext format
- Supports envelope encryption

TINK

- Safely implement common cryptographic tasks
- Widely deployed at Google
- AEAD, HMAC, key rotation
- Requires keys encrypted by KMS
- Encapsulates ciphertext format
- Supports envelope encryption
- Tooling for key configuration and rotation

TINK DEMO

PROBLEM: NO SCHEMA ENFORCEMENT

SOLUTION: JSON SCHEMA

JSON SCHEMA BASICS

JSON SCHEMA BASICS

- Annotate and validate JSON

JSON SCHEMA BASICS

- Annotate and validate JSON
- Describe data types and formats

JSON SCHEMA BASICS

- Annotate and validate JSON
- Describe data types and formats
- Generate human-readable documentation

JSON SCHEMA BASICS

- Annotate and validate JSON
- Describe data types and formats
- Generate human-readable documentation
- Composite types

JSON SCHEMA BASICS

- Annotate and validate JSON
- Describe data types and formats
- Generate human-readable documentation
- Composite types
- Composable multi-schema syntax

JSON SCHEMA BASICS

- Annotate and validate JSON
- Describe data types and formats
- Generate human-readable documentation
- Composite types
- Composable multi-schema syntax
- Extensible & customizable

USER TABLE

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
```

USER TABLE

```
CREATE TABLE users (
  id UUID    PRIMARY KEY,
  status     TEXT         NOT NULL CHECK (status IN ('enabled', 'di
```

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
  status      TEXT         NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
```

USER TABLE

```
CREATE TABLE users (
  id UUID    PRIMARY KEY,
  status     TEXT          NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
  created_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
```

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
  status      TEXT          NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
  created_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  username   VARCHAR(64)  NOT NULL UNIQUE,
```

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
  status      TEXT          NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
  created_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  username   VARCHAR(64)  NOT NULL UNIQUE,
  name       VARCHAR(128) NOT NULL,
```

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
  status      TEXT         NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
  created_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  username   VARCHAR(64)  NOT NULL UNIQUE,
  name       VARCHAR(128) NOT NULL,
  email      VARCHAR(254) NOT NULL CHECK (email ~* '^[A-Za-z0-9._+%
```

USER TABLE

```
CREATE TABLE users (
  id UUID     PRIMARY KEY,
  status      TEXT          NOT NULL CHECK (status IN ('enabled', 'di
  created_by VARCHAR(128) NOT NULL,
  updated_by VARCHAR(128) NOT NULL,
  created_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  updated_at TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
  username   VARCHAR(64)  NOT NULL UNIQUE,
  name       VARCHAR(128) NOT NULL,
  email      VARCHAR(254) NOT NULL CHECK (email ~* '^[A-Za-z0-9._+%
  phone      VARCHAR(15)  NOT NULL CHECK (phone ~* '^[+][0-9]{1,3}(
);
```

USER SCHEMA

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
    "status": {
      "description": "The status of the user in our systems",
      "type": "string",
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
    "status": {
      "description": "The status of the user in our systems",
      "type": "string",
      "default": "enabled",
      "enum": ["enabled", "disabled"]
    },
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
    "status": {
      "description": "The status of the user in our systems",
      "type": "string",
      "default": "enabled",
      "enum": ["enabled", "disabled"]
    },
    "profile": { "$ref": "profile.schema.json" }
  },
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
    "status": {
      "description": "The status of the user in our systems",
      "type": "string",
      "default": "enabled",
      "enum": ["enabled", "disabled"]
    },
    "profile": { "$ref": "profile.schema.json" }
  },
  "additionalProperties": false,
  "required": ["status"]
}
```

USER SCHEMA

```
{
  "title": "User",
  "description": "Information about a user.",
  "type": "object",
  "properties": {
    "status": {
      "description": "The status of the user in our systems",
      "type": "string",
      "default": "enabled",
      "enum": ["enabled", "disabled"]
    },
    "profile": { "$ref": "profile.schema.json" }
  },
  "additionalProperties": false,
  "required": ["status"]
}
```

PROFILE SCHEMA

PROFILE SCHEMA

```
{
  "title": "Profile",
  "description": "The user’s profile information",
  "type": "object",
```

PROFILE SCHEMA

```
{
  "title": "Profile",
  "description": "The user’s profile information",
  "type": "object",
  "properties": {
    "name": {
      "description": "The full/formatted name of the user",
      "type": "string",
      "maxLength": 128
    },
```

PROFILE SCHEMA

```
{
  "title": "Profile",
  "description": "The user’s profile information",
  "type": "object",
  "properties": {
    "name": {
      "description": "The full/formatted name of the user",
      "type": "string",
      "maxLength": 128
    },
    "username": {
      "description": "A unique username, to be used as a handle to reference th
      "type": "string",
      "maxLength": 64,
      "examples": ["theory", "strongrrl"]
    },
    "type": "string",
    "maxLength": 128
  },
  "username": {
    "description": "A unique username, to be used as a handle to reference th
    "type": "string",
    "maxLength": 64,
    "examples": ["theory", "strongrrl"]
  },
  "email": {
    "description": "The user’s email address.",
    "type": "string",
    "format": "email"
  },
      "maxLength": 128
    },
    "username": {
      "description": "A unique username, to be used as a handle to reference th
      "type": "string",
      "maxLength": 64,
      "examples": ["theory", "strongrrl"]
    },
    "email": {
      "description": "The user’s email address.",
      "type": "string",
      "format": "email"
    },
    "phone": {
      "description": "The user’s phone number",
      "type": "string",
      "pattern": "^[+][0-9]{1,3}(?:-[0-9]{2,6}){2,4}$",
      "examples": ["+1-212-555-0123", "+44-113-496-0123", "+353-020-919-1234"]
    }
  }
}
```

ENTITY SCHEMA

ENTITY SCHEMA

```
{
  "title": "Entity",
  "description": "Different types of entities and their data.",
  "type": "object",
  "properties": {
```

ENTITY SCHEMA

```
{
  "title": "Entity",
  "description": "Different types of entities and their data.",
  "type": "object",
  "properties": {
    "body": {
      "description": "Any number of different types of entity.",
      "anyOf": [
        { "$ref": "user.schema.json" }
        { "$ref": "organization.schema.json" }
      ]
    },
```

ENTITY SCHEMA

```
{
  "title": "Entity",
  "description": "Different types of entities and their data.",
  "type": "object",
  "properties": {
    "body": {
      "description": "Any number of different types of entity.",
      "anyOf": [
        { "$ref": "user.schema.json" }
        { "$ref": "organization.schema.json" }
      ]
    },
    "head": {
      "$ref": "head.schema.json"
    }
  }
}
```

HEAD SCHEMA

HEAD SCHEMA

```
{
  "title": "Head",
  "description": "Metadata exclusively managed by the server.",
  "type": "object",
  "readOnly": true,
```

HEAD SCHEMA

```
{
  "title": "Head",
  "description": "Metadata exclusively managed by the server.",
  "type": "object",
  "readOnly": true,
```

HEAD SCHEMA

```
{
  "title": "Head",
  "description": "Metadata exclusively managed by the server.",
  "type": "object",
  "readOnly": true,
  "properties": {
    "id": {
      "description": "The unique identifier for the entity",
      "type": "string",
      "$comment": "Base58-encoded UUID",
      "pattern": "^[123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz]
    },
```

HEAD SCHEMA

```
{
  "title": "Head",
  "description": "Metadata exclusively managed by the server.",
  "type": "object",
  "readOnly": true,
  "properties": {
    "id": {
      "description": "The unique identifier for the entity",
      "type": "string",
      "$comment": "Base58-encoded UUID",
      "pattern": "^[123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz]
    },
    "type": {
      "description": "The type of the entity",
      "type": "string",
      "$comment": "Must be kept in sync with the list of schemas in entity.body
      "enum": ["user", "organization"]
    "created_at": {
      "description": "Timestamp for when the entity was created",
      "type": "string",
      "format": "date-time"
    },
    "updated_at": {
      "description": "Timestamp indicating when the user entity was last update
      "type": "string",
      "format": "date-time"
    },
  "created_at": {
    "description": "Timestamp for when the entity was created",
    "type": "string",
    "format": "date-time"
  },
  "updated_at": {
    "description": "Timestamp indicating when the user entity was last update
    "type": "string",
    "format": "date-time"
  },
  "created_by": {
    "description": "Name of the client that created the entity.",
    "type": "string",
    "maxLength": 128,
    "examples": ["spiffe://nyt.net/id/lire"]
  },
  "updated_by": {
    "description": "Name of the client that last updated the entity.",
    "type": "string",
    "maxLength": 128,
    "examples": ["spiffe://nyt.net/subs/billing"]
  }
```

SCHEMA DEMO

FOSTER CONTINUITY

FOSTER CONTINUITY

- Goal: Document system of record

FOSTER CONTINUITY

- Goal: Document system of record
- Challenge: Entities scattered everywhere

FOSTER CONTINUITY

- Goal: Document system of record
- Challenge: Entities scattered everywhere
- Need to integrate data from partner teams

FOSTER CONTINUITY

- Goal: Document system of record
- Challenge: Entities scattered everywhere
- Need to integrate data from partner teams
- JSON Schema to the rescue!

FOSTER CONTINUITY

- Goal: Document system of record
- Challenge: Entities scattered everywhere
- Need to integrate data from partner teams
- JSON Schema to the rescue!
- Extensions sub-schema

EXTENSION SCHEMA

EXTENSION SCHEMA

```
{
  "title": "Extension",
  "description": "A extension with data useful to a specific business case.",
  "type": "object",
  "minProperties": 1,
```

EXTENSION SCHEMA

```
{
  "title": "Extension",
  "description": "A extension with data useful to a specific business case.",
  "type": "object",
  "minProperties": 1,
  "properties": {
    "id": {
      "description": "Extension’s object identifier, expected to be unique.",
      "type": "string"
    },
```

EXTENSION SCHEMA

```
{
  "title": "Extension",
  "description": "A extension with data useful to a specific business case.",
  "type": "object",
  "minProperties": 1,
  "properties": {
    "id": {
      "description": "Extension’s object identifier, expected to be unique.",
      "type": "string"
    },
    "@id": {
      "description": "JSON-LD URL for canonical resource the object",
      "type": "string",
      "format": "url"
    }
  },
```

EXTENSION SCHEMA

```
{
  "title": "Extension",
  "description": "A extension with data useful to a specific business case.",
  "type": "object",
  "minProperties": 1,
  "properties": {
    "id": {
      "description": "Extension’s object identifier, expected to be unique.",
      "type": "string"
    },
    "@id": {
      "description": "JSON-LD URL for canonical resource the object",
      "type": "string",
      "format": "url"
    }
  },
  "additionalProperties": true,
}
```

EXTENSIONS SCHEMA

EXTENSIONS SCHEMA

```
{
  "title": "Extensions",
  "description": "Extensions for varying business cases.",
  "type": "object",
  "minProperties": 1,
```

EXTENSIONS SCHEMA

```
{
  "title": "Extensions",
  "description": "Extensions for varying business cases.",
  "type": "object",
  "minProperties": 1,
  "propertyNames": {
    "$comment": "Should list all allowed extensions.",
    "enum": ["billing", "messaging", "games"]
  },
```

EXTENSIONS SCHEMA

```
{
  "title": "Extensions",
  "description": "Extensions for varying business cases.",
  "type": "object",
  "minProperties": 1,
  "propertyNames": {
    "$comment": "Should list all allowed extensions.",
    "enum": ["billing", "messaging", "games"]
  },
  "additionalProperties": { "$ref": "extension.schema.json" }
}
```

EXTENSIONS SCHEMA

Or team-specific schemas.

```
{
  "title": "Extensions",
  "description": "Extensions for varying business cases.",
  "type": "object",
  "minProperties": 1,
  "propertyNames": {
    "$comment": "Should list all allowed extensions.",
    "enum": ["billing", "messaging", "games"]
  },
  "additionalProperties": { "$ref": "extension.schema.json" }
}
```

EXTEND DEMO

HOLD UP

EXTENSION LOOKUP

EXTENSION LOOKUP

Clients need to fetch records by extension ID

EXTENSION LOOKUP

```
{
  "body": {
    "status": "enabled",
    "profile": {
      "username": "drummer",
      "name": "Camina Drummer",
      "email": "drummer@tycho.station"
    },
    "extensions": {
      "billing": {
        "id": "787ee95a8aec"
        "product": "home_delivery"
      }
    }
  }
}
```

EXTENSION LOOKUP

```
{
  "body": {
    "status": "enabled",
    "profile": {
      "username": "drummer",
      "name": "Camina Drummer",
      "email": "drummer@tycho.station"
    },
    "extensions": {
      "billing": {
        "id": "787ee95a8aec"
        "product": "home_delive
      }
    }
  }
}
```

Need to search secondary IDs!

TO THE DATABASE!

TO THE DATABASE!

```
id     | 0cf089f1-dd4d-45fb-8fd4-ec1e6120fc14
entity | \x01e991a23df43942b4c8db2f2e208c98060ca71fb400181a5
298b5c2c001bb64e30c89faf52856dd5de0b3a393342a026f154b67707c7
760d59ba4936791ede13a3d64306f3b9d67d96baa9aee26933e42f75f625
a3f20a38fc15f32676e34f35b4c53aa4dcbb24129a5e627b4b2dbdfb205e
fadd37a557f1103177d61d6fd0ae8e1dc926af4dfa9e5c67594912a10a53
36598b073aa7bd971f0819d3f8291b1fea0e3c8e6b08b37c9bdc5e9a1633
5b558058ccca05be8cc9d89af29be809846659556814871840e64b05a59e
7d3c06a79db84276fdcd3944bb521b88947ff513ae37c4851f4d3003b77c
bec3bacaf005a7af0f135031b2d3acaab620fcb46cb2990b6b645b0ad314
8a1e96b529c8bedfb34f85bab5d4d2b3187ba34eb7cdc96a4c442387c950
78c409dafbf767e3bb47861ae432fad
```

TO THE DATABASE!

Where is billing_id?

```
id     | 0cf089f1-dd4d-45fb-8fd4-ec1e6120fc14
entity | \x01e991a23df43942b4c8db2f2e208c98060ca71fb400181a5
298b5c2c001bb64e30c89faf52856dd5de0b3a393342a026f154b67707c7
760d59ba4936791ede13a3d64306f3b9d67d96baa9aee26933e42f75f625
a3f20a38fc15f32676e34f35b4c53aa4dcbb24129a5e627b4b2dbdfb205e
fadd37a557f1103177d61d6fd0ae8e1dc926af4dfa9e5c67594912a10a53
36598b073aa7bd971f0819d3f8291b1fea0e3c8e6b08b37c9bdc5e9a1633
5b558058ccca05be8cc9d89af29be809846659556814871840e64b05a59e
7d3c06a79db84276fdcd3944bb521b88947ff513ae37c4851f4d3003b77c
bec3bacaf005a7af0f135031b2d3acaab620fcb46cb2990b6b645b0ad314
8a1e96b529c8bedfb34f85bab5d4d2b3187ba34eb7cdc96a4c442387c950
78c409dafbf767e3bb47861ae432fad
```

ENCRYPTED DATA CANNOT BE QUERIED. 🤦

WHAT TO DO?

WHAT TO DO?

- 🔒 Encrypted data not searchable

WHAT TO DO?

- 🔒 Encrypted data not searchable
- 👿 Breached data must remain useless

WHAT TO DO?

- 🔒 Encrypted data not searchable
- 👿 Breached data must remain useless
- 💡 What if we hashed the data in a JSONB?

WHAT TO DO?

- 🔒 Encrypted data not searchable
- 👿 Breached data must remain useless
- 💡 What if we hashed the data in a JSONB?
- 🔑 Pair HMAC key to Tink AEAD key

WHAT TO DO?

- 🔒 Encrypted data not searchable
- 👿 Breached data must remain useless
- 💡 What if we hashed the data in a JSONB?
- 🔑 Pair HMAC key to Tink AEAD key
- 📇 Configure data to be indexed

SCHEMA REVISION

```
CREATE TABLE users (
    id      UUID PRIMARY KEY,
    entity BYTEA
);
```

SCHEMA REVISION

```
CREATE TABLE users (
    id     UUID PRIMARY KEY,
    entity BYTEA,
    LOOKUP JSONB
);

CREATE INDEX users_lookup_idx ON users
 USING GIN (lookup jsonb_path_ops);
```

SCHEMA REVISION

```
CREATE TABLE users (
    id     UUID PRIMARY KEY,
    entity BYTEA,
    LOOKUP JSONB
);

CREATE INDEX users_lookup_idx ON users
 USING GIN (lookup jsonb_path_ops);
```

KEYRING DEMO

SO HOW DO WE QUERY IT?

SQL/JSON PATH

SQL/JSON PATH

SQL/JSON PATH

- ✅ SQL 2016 Standard

SQL/JSON PATH

- ✅ SQL 2016 Standard
- 🐘 Implemented in PostgreSQL 12

SQL/JSON PATH

- ✅ SQL 2016 Standard
- 🐘 Implemented in PostgreSQL 12
- 🪣 Supports placeholder variables

SQL/JSON PATH

- ✅ SQL 2016 Standard
- 🐘 Implemented in PostgreSQL 12
- 🪣 Supports placeholder variables
- 👫 Query multiple values at once

WHERE → PATH

WHERE → PATH

```
WHERE profile.email = $1
```

WHERE → PATH

```
WHERE profile.email = $1
```

```
$.profile ?(@.email == $email)
```

WHERE → PATH

```
WHERE profile.email = $1
$.profile ?(@.email == $email)

WHERE status = 'enabled' AND profile.email = $1
```

WHERE → PATH

```
WHERE profile.email = $1
$.profile ?(@.email == $email)

WHERE status = 'enabled' AND profile.email = $1
$ ?(@.status = 'enabled' && @.profile.email == $email)
```

WHERE → PATH

```
WHERE profile.email = $1
$.profile ?(@.email == $email)

WHERE status = 'enabled' AND profile.email = $1
$ ?(@.status = 'enabled' && @.profile.email == $email)

WHERE profile.email = $1 OR profile.email2 = $1
```

WHERE → PATH

```
WHERE profile.email = $1
$.profile ?(@.email == $email)

WHERE status = 'enabled' AND profile.email = $1
$ ?(@.status = 'enabled' && @.profile.email == $email)

WHERE profile.email = $1 OR profile.email2 = $1
$.profile ?(@.email == $email || @.email2 == $email)
```

PATH CONFIG

PATH CONFIG

```
"lookups": {
```

PATH CONFIG

```
"lookups": {
  "email": "$.body.profile ?(@.email == $email)",
```

PATH CONFIG

```
"lookups": {
  "email": "$.body.profile ?(@.email == $email)",
  "username": "$.body.profile ?(@.username == $username)",
```

PATH CONFIG

```
"lookups": {
  "email": "$.body.profile ?(@.email == $email)",
  "username": "$.body.profile ?(@.username == $username)",
  "billing_id": "$.body.extensions.billing ?(@.id == $billing_id)"
}
```

PATH CONFIG

```
"lookups": {
  "email": "$.body.profile ?(@.email == $email)",
  "username": "$.body.profile ?(@.username == $username)",
  "billing_id": "$.body.extensions.billing ?(@.id == $billing_id)"
}
```

Also defines fields to index

SECONDARY KEY API

SECONDARY KEY API

```
GET /users/{key}:{value}
```

https://justatheory.com/2022/08/rfc-restful-secondary-key-api/

SECONDARY KEY API

```
GET /users/{key}:{value}

GET /users/email:hi@example.com
```

https://justatheory.com/2022/08/rfc-restful-secondary-key-api/

SECONDARY KEY API

```
GET /users/{key}:{value}

GET /users/email:hi@example.com

GET /users/username:theory
```

https://justatheory.com/2022/08/rfc-restful-secondary-key-api/

SECONDARY KEY API

```
GET /users/{key}:{value}

GET /users/email:hi@example.com

GET /users/username:theory

GET /users/billing_id:787ee95a8aec
```

https://justatheory.com/2022/08/rfc-restful-secondary-key-api/

API LOGIC

API LOGIC

```
GET /users/email:drummer@tycho.station
```

API LOGIC

```
GET /users/email:drummer@tycho.station

"email": "$.body.profile ?(@.email == $email)"
```

API LOGIC

```
GET /users/email:drummer@tycho.station

"email": "$.body.profile ?(@.email == $email)"

$.body.profile ?(@.email == "drummer@tycho.station")
```

API LOGIC

```
GET /users/email:drummer@tycho.station

"email": "$.body.profile ?(@.email == $email)"

$.body.profile ?(@.email == "drummer@tycho.station")

$."MHc3r"."yu8_f" ?(@."WO+iq" == "^iN)71kxTVbtWGY-6v-B%ad<^")
```

API LOGIC

```
GET /users/email:drummer@tycho.station

"email": "$.body.profile ?(@.email == $email)"

$.body.profile ?(@.email == "drummer@tycho.station")

$."MHc3r"."yu8_f" ?(@."WO+iq" == "^iN)71kxTVbtWGY-6v-B%ad<^")

$."X%OpZ"."QjC/c" ?(

    @."mL4eA" == "(og?ozk2(}6tj0XjhRtP/S^V{:08]bAc#gbt7Y)+"

)
```

RESULTING QUERY

RESULTING QUERY

```
SELECT id, entity
  FROM demo.users
 WHERE lookup @? ANY(ARRAY[
     '$."MHc3r"."yu8_f" ?(@."WO+iq" == "^iN)71kxTVb
     '$."X%OpZ"."QjC/c" ?(@."mL4eA" == "(og?ozk2(}6
 ]);
```

RESULTING QUERY

```
SELECT id, entity
  FROM demo.users
 WHERE lookup @? ANY(ARRAY[
    '$."MHc3r"."yu8_f" ?(@."WO+iq" == "^iN)71kxTVb
    '$."X%OpZ"."QjC/c" ?(@."mL4eA" == "(og?ozk2(}6
 ]);
```

HOW TO HASH?

SQL/JSON PATH PORT

SQL/JSON PATH PORT

- Need to parse JSON Paths

SQL/JSON PATH PORT

- Need to parse JSON Paths
    - Replace keys with truncated hashes

SQL/JSON PATH PORT

- Need to parse JSON Paths
  - Replace keys with truncated hashes
  - Replace variables & values with hashes

SQL/JSON PATH PORT

- Need to parse JSON Paths
  - Replace keys with truncated hashes
  - Replace variables & values with hashes
  - Must understand path syntax

SQL/JSON PATH PORT

- Need to parse JSON Paths
  - Replace keys with truncated hashes
  - Replace variables & values with hashes
  - Must understand path syntax
- Service written in Go

SQL/JSON PATH PORT

- Need to parse JSON Paths
  - Replace keys with truncated hashes
  - Replace variables & values with hashes
  - Must understand path syntax
- Service written in Go
- Ported SQL/JSON Path parser to Go!

QUERY HASHER

QUERY HASHER

```
func (m *Model) queries(path string, params map[string]any) []string {
```

QUERY HASHER

```go
func (m *Model) queries(path string, params map[string]any) []string {
    ast := path.Parse(path)
```

QUERY HASHER

```go
func (m *Model) queries(path string, params map[string]any) []string {
    ast := path.Parse(path)

    hashers := m.dek.Hashers()
    queries := make([]string, len(hashers))
```

QUERY HASHER

```go
func (m *Model) queries(path string, params map[string]any) []string {
    ast := path.Parse(path)

    hashers := m.dek.Hashers()
    queries := make([]string, len(hashers))
    for i, h := range hashers {
        queries[i] = ast.HashCompile(h, params)
    }
```

QUERY HASHER

```go
func (m *Model) queries(path string, params map[string]any) []string {
    ast := path.Parse(path)

    hashers := m.dek.Hashers()
    queries := make([]string, len(hashers))
    for i, h := range hashers {
        queries[i] = ast.HashCompile(h, params)
    }
    return queries
}
```

QUERY HASHER

```go
func (m *Model) queries(path string, params map[string]any) []string {
    ast := path.Parse(path)

    hashers := m.dek.Hashers()
    queries := make([]string, len(hashers))
    for i, h := range hashers {
        queries[i] = ast.HashCompile(h, params)
    }
    return queries
}

func (ast *AST) HashCompile(h Hasher, params [string]any) string {
    return ast.hashNode(ast.root, h, params).String()
}
```

NODE HASHER

NODE HASHER

```
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
```

NODE HASHER

```go
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
    switch n := n.(type) {
```

NODE HASHER

```go
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
    switch n := n.(type) {
    case *String, *Number, *Bool, *Null:
        return NewString(h.Hash(n.Text()))
```

NODE HASHER

```
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
    switch n := n.(type) {
    case *String, *Number, *Bool, *Null:
        return NewString(h.Hash(n.Text()))
    case *Key:
        return NewKey(h.Hash(n.Text())[0:4])
```

NODE HASHER

```go
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
    switch n := n.(type) {
    case *String, *Number, *Bool, *Null:
        return NewString(h.Hash(n.Text()))
    case *Key:
        return NewKey(h.Hash(n.Text())[0:4])
    case *Variable:
        return NewString(h.Hash(params[n.Text()]))
```

NODE HASHER

```go
func (ast *AST) hashNode(n *Node, h Hasher, params [string]any) *Node {
    switch n := n.(type) {
    case *String, *Number, *Bool, *Null:
        return NewString(h.Hash(n.Text()))
    case *Key:
        return NewKey(h.Hash(n.Text())[0:4])
    case *Variable:
        return NewString(h.Hash(params[n.Text()]))
    case *Unary:
        arg := ast.hashNode(n.Arg(), h, params)
        return ast.NewUnary(n.Type(), arg)
   }
}
```

LOOKUP DEMO

DOWNSIDES

DOWNSIDES

- Key rotation requires record rotation

DOWNSIDES

- Key rotation requires record rotation
- Schema changes don’t validate existing records

DOWNSIDES

- Key rotation requires record rotation
- Schema changes don’t validate existing records
- Existing data not indexed for new lookup fields

DOWNSIDES

- Key rotation requires record rotation
- Schema changes don’t validate existing records
- Existing data not indexed for new lookup fields
- No secondary key unique enforcement

UPKEEPER

FUTURE WORK

FUTURE WORK

- Upkeeper

FUTURE WORK

- Upkeeper
- SQL/JSON Parser rewrite

FUTURE WORK

- Upkeeper
- SQL/JSON Parser rewrite
- First class Tink Key integration

FUTURE WORK

- Upkeeper
- SQL/JSON Parser rewrite
- First class Tink Key integration
- Unencrypted Head column?

FUTURE WORK

- Upkeeper
- SQL/JSON Parser rewrite
- First class Tink Key integration
- Unencrypted Head column?
- Open Source?

THANK YOU

Cipher Doc / David E. Wheeler / PGCon 2023
david.wheeler@nytimes.com
justatheory.com