New issue

[Feature Suggestion]: Change / Pass Workspace Name in PgSQL Storages #1244

Open

Labels question

0bserver07 opened on Apr 1 · edited by 0bserver07 Edits

Would you folks like me to do a PR to fix this feature?

Currently every PgStorage "workspace" is set to default, regardless of changing the config in the LightRAG Client

I'll explain how I will modify the code to implement approach. First, let me show you the changes we need to make:

```
class ClientManager:
    _instances: dict[str, Any] = {"db": None, "ref_count": 0}
    _lock = asyncio.Lock()

    @staticmethod
    def get_config(global_config: dict[str, Any] | None = None) -> dict[str, Any]:
        # First try to get workspace from global config
        workspace = None
        if global_config and "vector_db_storage_cls_kwargs" in global_config:
            workspace = global_config["vector_db_storage_cls_kwargs"].get("workspace")

        # Read standard config
        config = configparser.ConfigParser()
        config.read("config.ini", "utf-8")

        return {
            "host": os.environ.get(
                "POSTGRES_HOST",
                config.get("postgres", "host", fallback="localhost"),
            ),
            "port": os.environ.get(
                "POSTGRES_PORT", config.get("postgres", "port", fallback=5432)
            ),
            "user": os.environ.get(
                "POSTGRES_USER",
                config.get("postgres", "user", fallback=None),
            ),
```

```
            "password": os.environ.get(
                "POSTGRES_PASSWORD",
                config.get("postgres", "password", fallback=None),
            ),
            "database": os.environ.get(
                "POSTGRES_DATABASE",
                config.get("postgres", "database", fallback=None),
            ),
            # Use workspace from global config if available, otherwise fall back to env/config.in
            "workspace": workspace or os.environ.get(
                "POSTGRES_WORKSPACE",
                config.get("postgres", "workspace", fallback="default"),
            ),
        }

    @classmethod
    async def get_client(cls, global_config: dict[str, Any] | None = None) -> PostgreSQLDB:
        async with cls._lock:
            if cls._instances["db"] is None:
                config = cls.get_config(global_config)
                db = PostgreSQLDB(config)
                await db.initdb()
                await db.check_tables()
                cls._instances["db"] = db
                cls._instances["ref_count"] = 0
            cls._instances["ref_count"] += 1
            return cls._instances["db"]
```

Now we need to modify the storage classes to pass their global config to the ClientManager. Let me update those:

```
@final
@dataclass
class PGKVStorage(BaseKVStorage):
    db: PostgreSQLDB = field(default=None)

    def __post_init__(self):
        namespace_prefix = self.global_config.get("namespace_prefix")
        self.base_namespace = self.namespace.replace(namespace_prefix, "")
        self._max_batch_size = self.global_config["embedding_batch_num"]

    async def initialize(self):
        if self.db is None:
            self.db = await ClientManager.get_client(self.global_config)

// ... existing code ...

@final
@dataclass
class PGVectorStorage(BaseVectorStorage):
    db: PostgreSQLDB | None = field(default=None)

    def __post_init__(self):
        self._max_batch_size = self.global_config["embedding_batch_num"]
        namespace_prefix = self.global_config.get("namespace_prefix")
        self.base_namespace = self.namespace.replace(namespace_prefix, "")
        config = self.global_config.get("vector_db_storage_cls_kwargs", {})
```

```
            cosine_threshold = config.get("cosine_better_than_threshold")
            if cosine_threshold is None:
                raise ValueError(
                    "cosine_better_than_threshold must be specified in vector_db_storage_cls_kwargs"
                )
            self.cosine_better_than_threshold = cosine_threshold

        async def initialize(self):
            if self.db is None:
                self.db = await ClientManager.get_client(self.global_config)

    // ... existing code ...

    @final
    @dataclass
    class PGDocStatusStorage(DocStatusStorage):
        db: PostgreSQLDB = field(default=None)

        async def initialize(self):
            if self.db is None:
                self.db = await ClientManager.get_client(self.global_config)
```

Here's what these changes do:

1. In `ClientManager`:
   - Added `global_config` parameter to `get_config` and `get_client` methods
   - The `get_config` method now first checks for workspace in the global config's `vector_db_storage_cls_kwargs`
   - If found, uses that workspace; otherwise falls back to environment variables and config.ini
2. In the storage classes (`PGKVStorage`, `PGVectorStorage`, `PGDocStatusStorage`):
   - Modified `initialize` methods to pass their `global_config` to `ClientManager.get_client()`
   - This ensures the workspace setting from your configuration gets used

Now when you run LightRAG client:

1. The workspace from can be set in the config
2. This config flows through LightRAG to the storage classes
3. The storage classes pass it to ClientManager
4. ClientManager uses this workspace value instead of falling back to default

This should make PostgreSQL respect the workspace setting you provide in your configuration.

Additional Context

```
+++ b/lightrag/kg/postgres_impl.py
@@ -192,7 +192,13 @@ class ClientManager:
     _lock = asyncio.Lock()

     @staticmethod
```

```
-    def get_config() -> dict[str, Any]:
+    def get_config(global_config: dict[str, Any] | None = None) -> dict[str, Any]:
+        # First try to get workspace from global config
+        workspace = None
+        if global_config and "vector_db_storage_cls_kwargs" in global_config:
+            workspace = global_config["vector_db_storage_cls_kwargs"].get("workspace")
+
+        # Read standard config
         config = configparser.ConfigParser()
         config.read("config.ini", "utf-8")

@@ -205,7 +211,8 @@ class ClientManager:
                 "POSTGRES_PORT", config.get("postgres", "port", fallback=5432)
             ),
             "user": os.environ.get(
-                "POSTGRES_USER", config.get("postgres", "user", fallback=None)
+                "POSTGRES_USER",
+                config.get("postgres", "user", fallback=None),
             ),
             "password": os.environ.get(
                 "POSTGRES_PASSWORD",
@@ -215,17 +222,18 @@ class ClientManager:
                 "POSTGRES_DATABASE",
                 config.get("postgres", "database", fallback=None),
             ),
-            "workspace": os.environ.get(
+            # Use workspace from global config if available, otherwise fall back to env/config.i
+            "workspace": workspace or os.environ.get(
                 "POSTGRES_WORKSPACE",
                 config.get("postgres", "workspace", fallback="default"),
             ),
         }

     @classmethod
-    async def get_client(cls) -> PostgreSQLDB:
+    async def get_client(cls, global_config: dict[str, Any] | None = None) -> PostgreSQLDB:
         async with cls._lock:
             if cls._instances["db"] is None:
-                config = ClientManager.get_config()
+                config = cls.get_config(global_config)
                 db = PostgreSQLDB(config)
                 await db.initdb()
                 await db.check_tables()
@@ -260,7 +268,7 @@ class PGKVStorage(BaseKVStorage):

     async def initialize(self):
         if self.db is None:
-            self.db = await ClientManager.get_client()
+            self.db = await ClientManager.get_client(self.global_config)

     async def finalize(self):
         if self.db is not None:
@@ -405,7 +413,7 @@ class PGVectorStorage(BaseVectorStorage):

     async def initialize(self):
         if self.db is None:
-            self.db = await ClientManager.get_client()
+            self.db = await ClientManager.get_client(self.global_config)

     async def finalize(self):
```

```
         if self.db is not None:
@@ -698,7 +706,7 @@ class PGDocStatusStorage(DocStatusStorage):

     async def initialize(self):
         if self.db is None:
-            self.db = await ClientManager.get_client()
         config.read("config.ini", "utf-8")

@@ -205,7 +211,8 @@ class ClientManager:
                 "POSTGRES_PORT", config.get("postgres", "port", fallback=5432)
             ),
             "user": os.environ.get(
-                "POSTGRES_USER", config.get("postgres", "user", fallback=None)
+                "POSTGRES_USER",
+                config.get("postgres", "user", fallback=None),
             ),
             "password": os.environ.get(
                 "POSTGRES_PASSWORD",
@@ -215,17 +222,18 @@ class ClientManager:
                 "POSTGRES_DATABASE",
                 config.get("postgres", "database", fallback=None),
             ),
-            "workspace": os.environ.get(
+            # Use workspace from global config if available, otherwise fall back to env/config.i
+            "workspace": workspace or os.environ.get(
                 "POSTGRES_WORKSPACE",
                 config.get("postgres", "workspace", fallback="default"),
             ),
         }

     @classmethod
-    async def get_client(cls) -> PostgreSQLDB:
+    async def get_client(cls, global_config: dict[str, Any] | None = None) -> PostgreSQLDB:
         async with cls._lock:
             if cls._instances["db"] is None:
-                config = ClientManager.get_config()
+                config = cls.get_config(global_config)
                 db = PostgreSQLDB(config)
                 await db.initdb()
                 await db.check_tables()
@@ -260,7 +268,7 @@ class PGKVStorage(BaseKVStorage):

     async def initialize(self):
         if self.db is None:
-            self.db = await ClientManager.get_client()
+            self.db = await ClientManager.get_client(self.global_config)

     async def finalize(self):
         if self.db is not None:
@@ -405,7 +413,7 @@ class PGVectorStorage(BaseVectorStorage):

     async def initialize(self):
         if self.db is None:
-            self.db = await ClientManager.get_client()
+            self.db = await ClientManager.get_client(self.global_config)

     async def finalize(self):
         if self.db is not None:
@@ -698,7 +706,7 @@ class PGDocStatusStorage(DocStatusStorage):
```

```
    async def initialize(self):
        if self.db is None:
-           self.db = await ClientManager.get_client()
+           self.db = await ClientManager.get_client(self.global_config)

    async def finalize(self):
        if self.db is not None:
```

👍 1

0bserver07 added **question** on Apr 1

0bserver07 changed the title ~~[Question]: Change / Pass Workspace Name in PgSQL Storages~~ [Feature Request]: Change / Pass Workspace Name in PgSQL Storages on Apr 1

0bserver07 changed the title ~~[Feature Request]: Change / Pass Workspace Name in PgSQL Storages~~ [Feature Suggestion]: Change / Pass Workspace Name in PgSQL Storages on Apr 1

Metadata

Assignees

No one assigned

Labels

question

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants