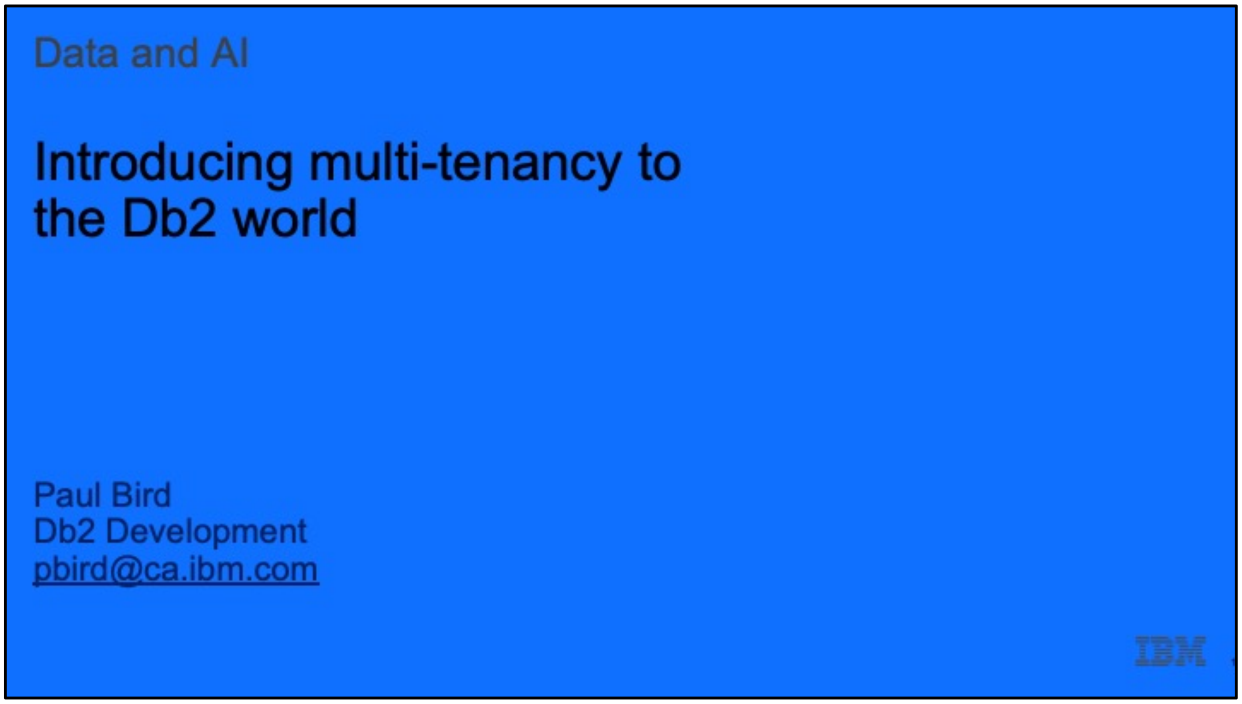


Data and AI

Introducing multi-tenancy to the Db2 world

Paul Bird
Db2 Development
pbird@ca.ibm.com

IBM

Have you ever struggled with schema collision when trying to consolidate different databases? Do you wish you could have a separate, independent copy of your application(s) on the same hardware as your production database just for QA testing? This session reveals the new capabilities being introduced to the Db2 family of products to help deal with these issues both now and in the future.

Paul Bird is a senior technical staff member (STSM) in the Db2 development organization. For the last 30 years, he has worked on the inside of the DB2 for Linux, Unix, and Windows product as a lead developer and architect with a focus on such diverse areas as workload management, monitoring, security, upgrade, JSON, and general SQL processing. You can reach him at pbird@ca.ibm.com.

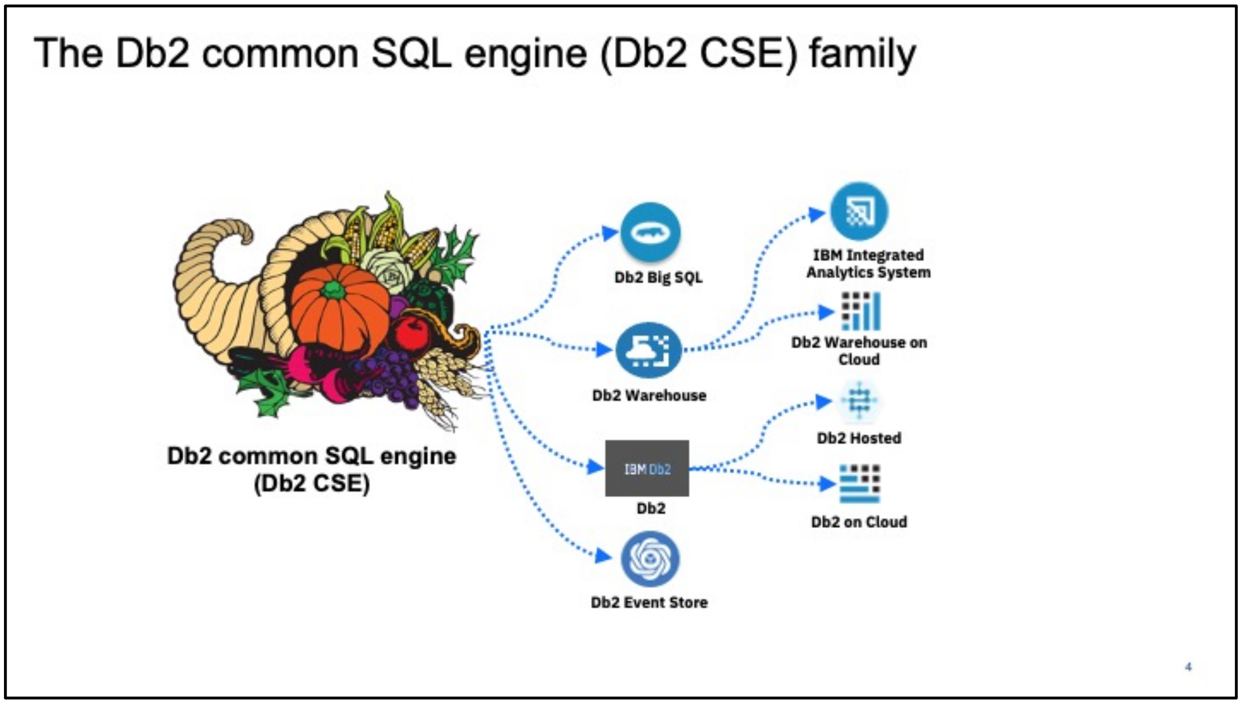
Agenda

Learn about the new tenant concept being introduced into the Db2 common SQL engine (Db2 CSE)

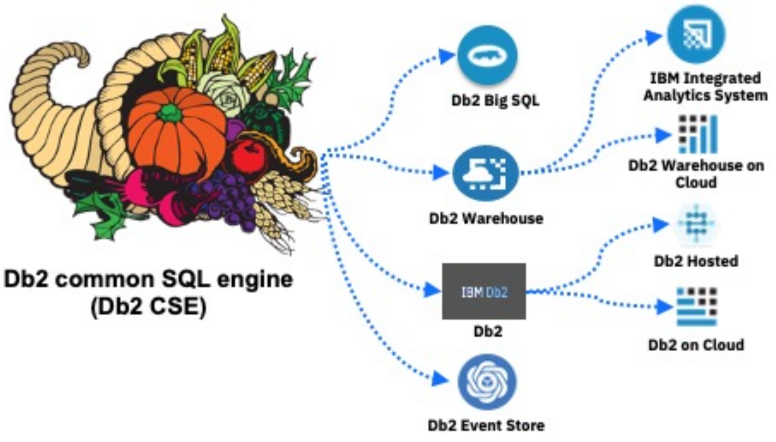
Understand the initial tenant capabilities being delivered

Gain insight (and provide feedback) on the planned roadmap

Please note :



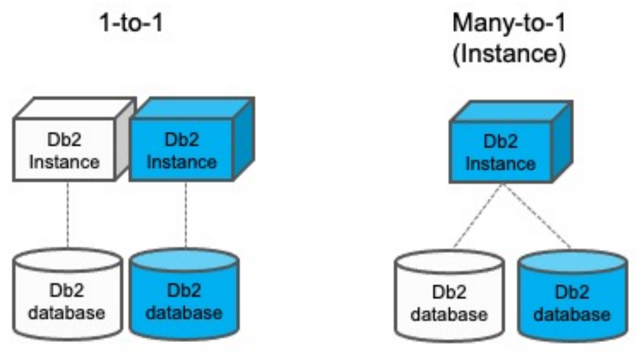
The Db2 common SQL engine (Db2 CSE) family



All of these Db2 family of products share the same underlying common SQL engine within the different Db2 form factors. The same Db2 code base provides the underlying database technology used by the Db2 software product and its additional form factors as well as the Db2 Warehouse offering and its additional ones. The Db2 Big SQL product and Db2 Event Store products also are embraced by this strategy.

It is the existence of this common layer that enables Db2 to support the “Write once, run anywhere” objective.

Traditional Db2 database “multi-tenancy” configurations



What is (database) multi-tenancy?

A single physical database supporting multiple "tenants" where:

– All tenants share the same database infrastructure and resources

– Each tenant has a private, isolated perspective for their own data

Multi-tenancy viewed as an apartment building

A common analogy used for database multi-tenancy is that of an apartment building where the physical database is the apartment building and the individual tenants defined in that database are the individual apartments. Within their individual apartments, occupants have complete autonomy over the apartment contents (and paint colours!) but they share the hallways and building infrastructure with each other. While they have privacy in their own rooms, their actions (such as flushing the toilet or playing loud music) can have impact on the other tenants.

Why database multi-tenancy is interesting

Cost-savings through consolidation (“One database versus many”)

- Reduction of fixed overhead costs associated with individual databases
- Centralization/simplification of database operations

Sharing unique environments without collisions

- Development environments are rarely at the same scale as production environments

Why is Db2 looking at database multi-tenancy?

Significant cost-savings for our customers with many small development systems

Netezza/Postgres compatibility

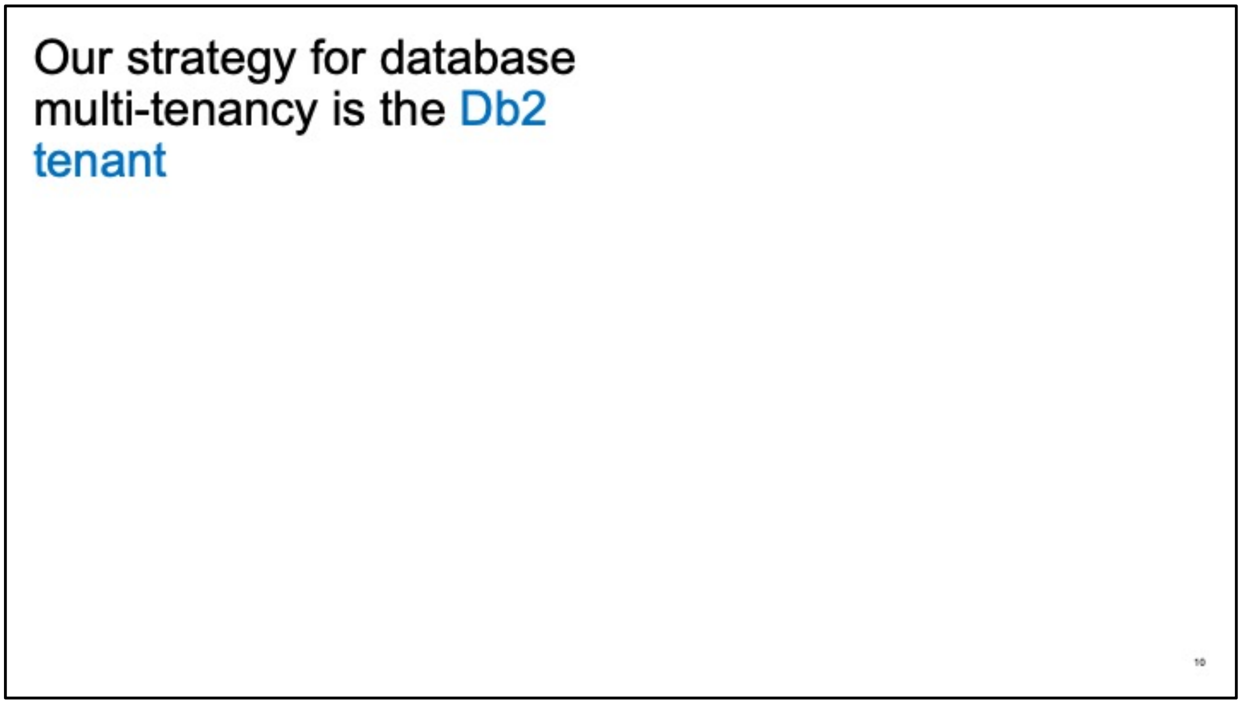
Customer and IBM cloud offerings

– Multi-tenancy is a natural offshoot of multi-user cloud offerings

Provides workaround for those configurations where more than 1 database is not possible:

– Currently, Db2 warehouse and IIAS products only allow 1 database

– SAP on DB2 pureScale configurations effectively only allow 1 database

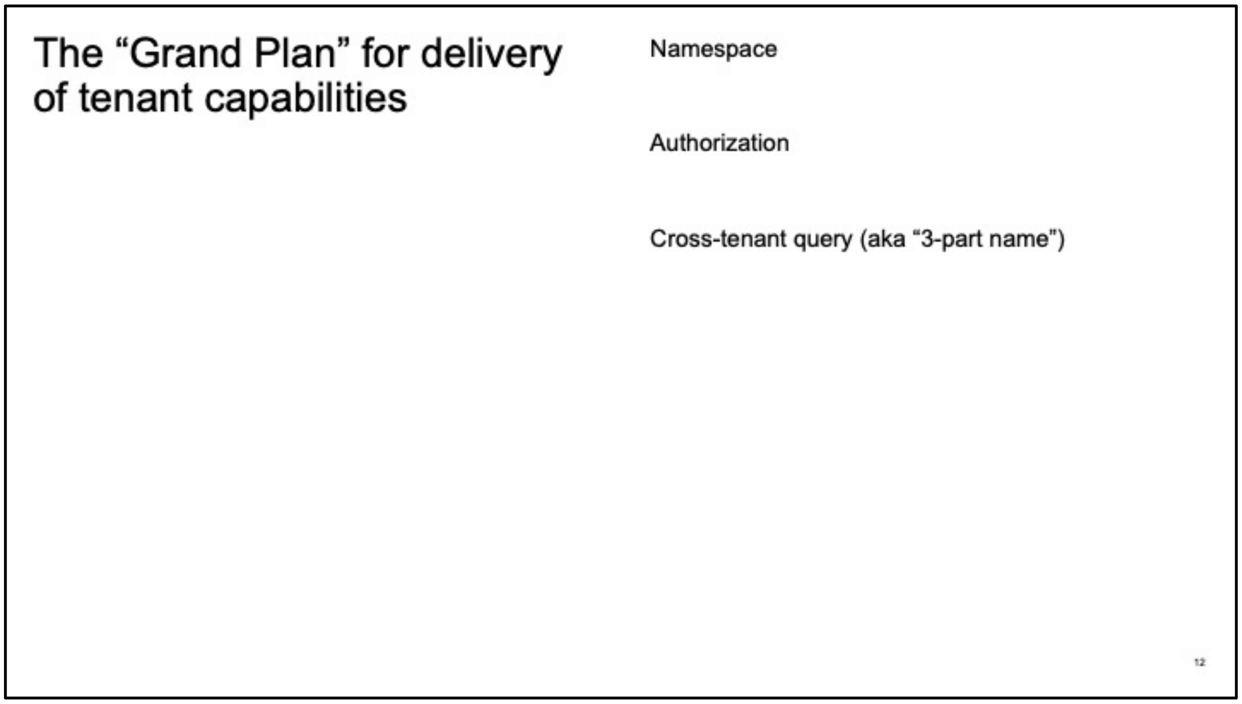


Our strategy for database multi-tenancy is the Db2 tenant

Introducing the Db2 tenant

A new database object

A set of capabilities to be shipped over time in the Db2 CSE codebase



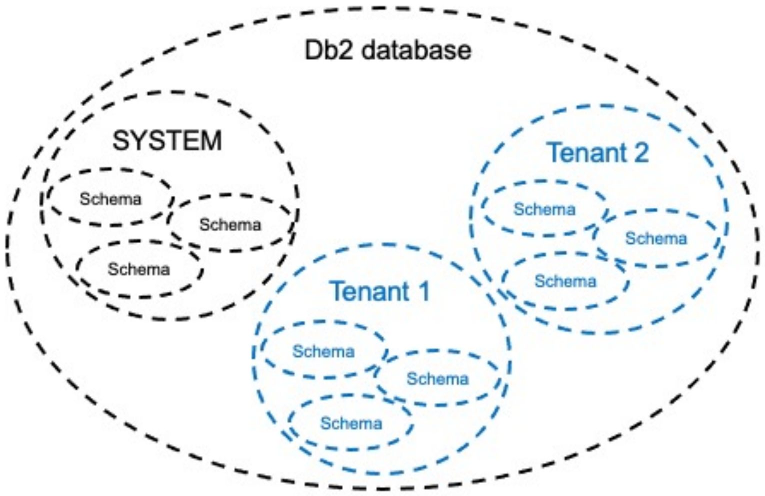
The “Grand Plan” for delivery of tenant capabilities

Namespace

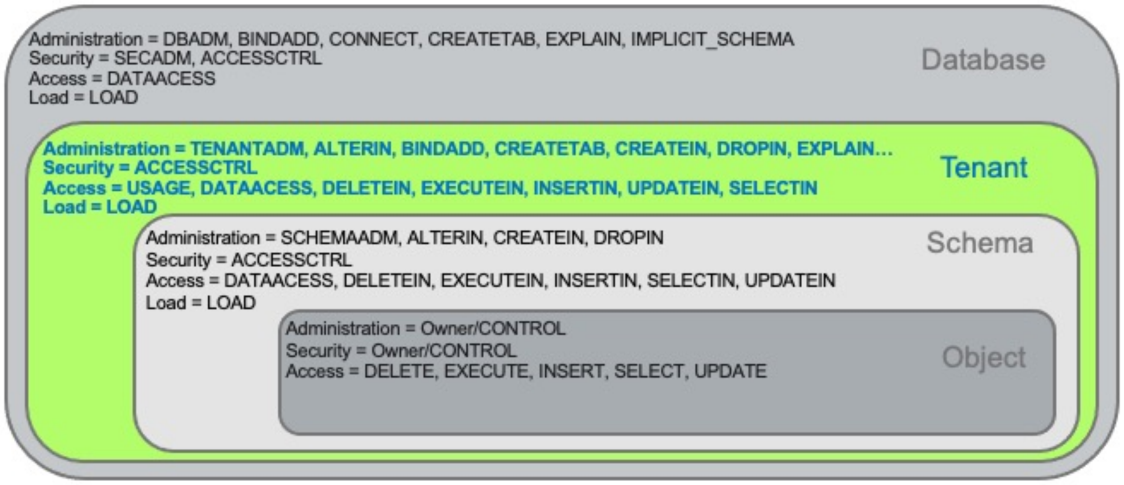
Authorization

Cross-tenant query (aka “3-part name”)

Namespace


Db2 database
SYSTEM
Schema
Schema
Schema
Tenant 1
Schema
Schema
Schema
Tenant 2
Schema
Schema
Schema

Authorization



Administration = DBADM, BINDADD, CONNECT, CREATETAB, EXPLAIN, IMPLICIT_SCHEMA
Security = SECADM, ACCESSCTRL
Access = DATAACESS
Load = LOAD

Administration = TENANTADM, ALTERIN, BINDADD, CREATETAB, CREATEIN, DROPIN, EXPLAIN…
Security = ACCESSCTRL
Access = USAGE, DATAACESS, DELETEIN, EXECUTEIN, INSERTIN, UPDATEIN, SELECTIN
Load = LOAD

Administration = SCHEMAADM, ALTERIN, CREATEIN, DROPIN
Security = ACCESSCTRL
Access = DATAACESS, DELETEIN, EXECUTEIN, INSERTIN, SELECTIN, UPDATEIN
Load = LOAD

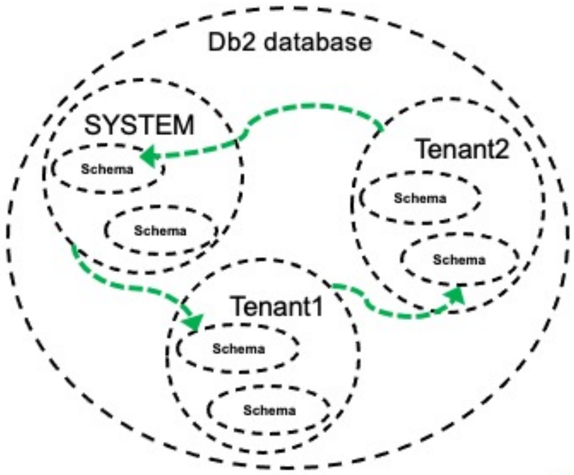
Administration = Owner/CONTROL
Security = Owner/CONTROL
Access = DELETE, EXECUTE, INSERT, SELECT, UPDATE

Cross-tenant query

The ability for DML statements to reference table objects in other tenants

E.g., while in Tenant 1, be able to issue:

– SELECT * FROM TENANT2.PBIRD.T1



Initial “Namespace” delivery

The objectives

Within the same Db2 CSE database, we want to be able to give different users/applications:

- The ability to have an independent catalog namespace, from the schema level and downwards, for user defined objects
  - Create and manipulate user-defined database objects
  - Insert, modify, and access data in user-defined database objects
- The ability to have distinct authorization scope for objects within these namespaces
  - Manage security permissions on user-defined objects

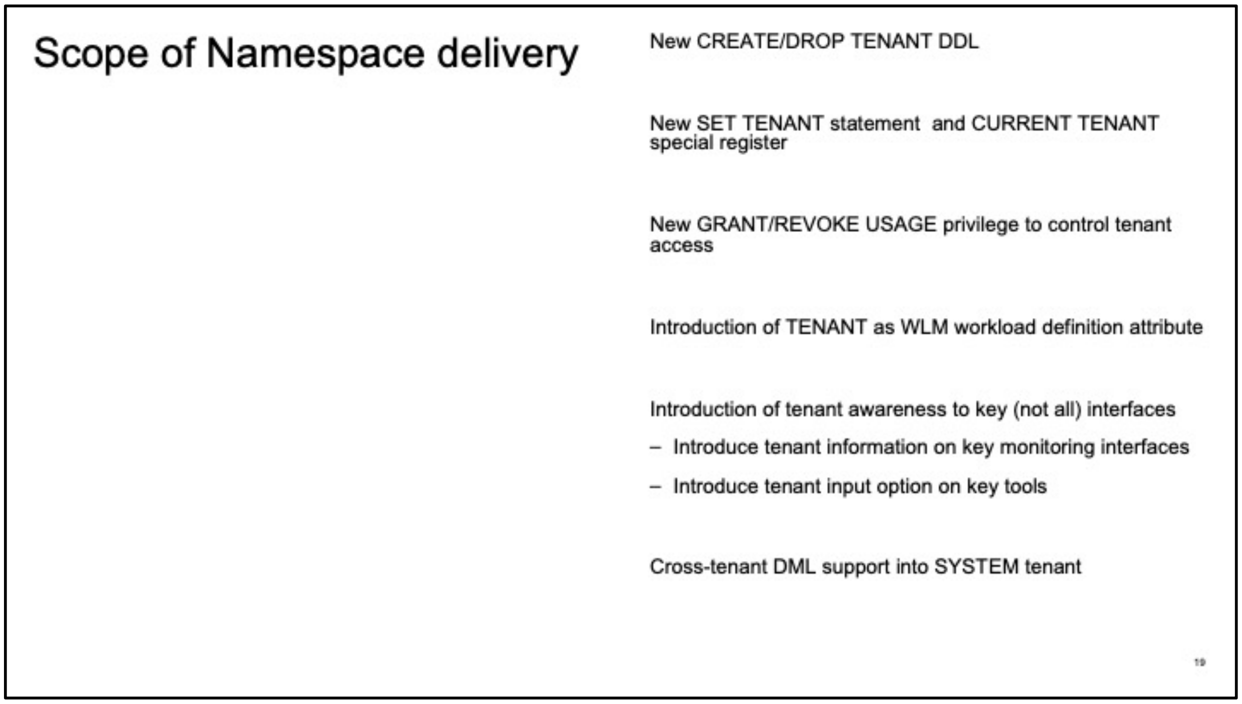
Target usage scenarios

Consolidation of smaller test systems

– Reduces physical footprint and simplify database operations while maintaining relative independence

Allow for (off-hour) development testing in production h/w environment without risk to production data or changes to user applications

– E.g., full-scale development testing using the same database on same machine



Scope of Namespace delivery

New CREATE/DROP TENANT DDL

New SET TENANT statement and CURRENT TENANT special register

New GRANT/REVOKE USAGE privilege to control tenant access

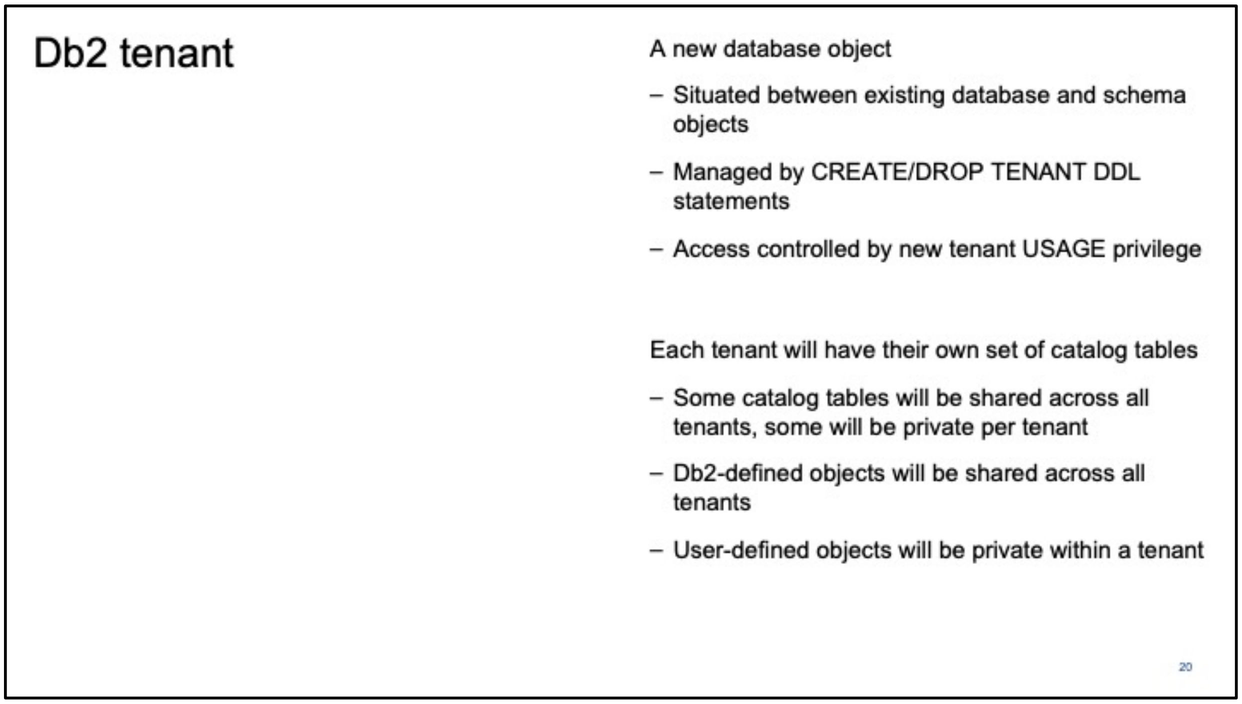
Introduction of TENANT as WLM workload definition attribute

Introduction of tenant awareness to key (not all) interfaces

– Introduce tenant information on key monitoring interfaces

– Introduce tenant input option on key tools

Cross-tenant DML support into SYSTEM tenant



Db2 tenant

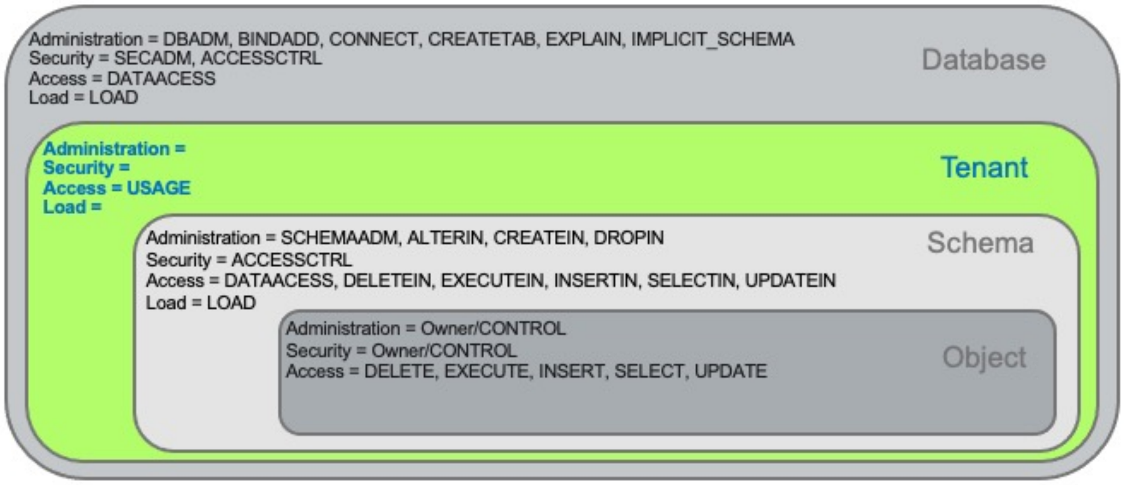
A new database object

- Situated between existing database and schema objects
- Managed by CREATE/DROP TENANT DDL statements
- Access controlled by new tenant USAGE privilege

Each tenant will have their own set of catalog tables

- Some catalog tables will be shared across all tenants, some will be private per tenant
- Db2-defined objects will be shared across all tenants
- User-defined objects will be private within a tenant

New tenant level privilege


Database
Administration = DBADM, BINDADD, CONNECT, CREATETAB, EXPLAIN, IMPLICIT_SCHEMA
Security = SECADM, ACCESSCTRL
Access = DATAACESS
Load = LOAD
Tenant
Administration =
Security =
Access = USAGE
Load =
Schema
Administration = SCHEMAADM, ALTERIN, CREATEIN, DROPIN
Security = ACCESSCTRL
Access = DATAACESS, DELETEIN, EXECUTEIN, INSERTIN, SELECTIN, UPDATEIN
Load = LOAD
Object
Administration = Owner/CONTROL
Security = Owner/CONTROL
Access = DELETE, EXECUTE, INSERT, SELECT, UPDATE

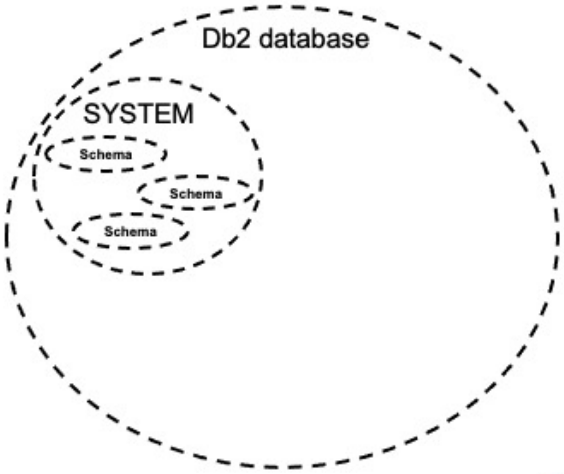
The default SYSTEM tenant

The initial set of catalogs established when the database is created

– This is referred to as the SYSTEM tenant

– Cannot be removed

– Contains catalog information for shared resources and Db2 defined objects



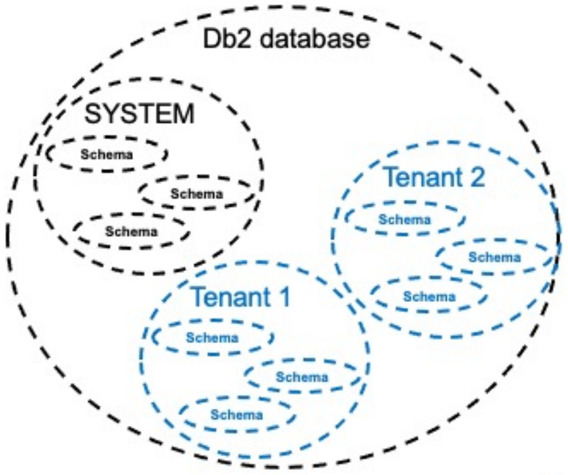
22

User-defined tenants

A DBADM can create a tenant to set up an independent catalog namespace within a Db2 database

All connections made to a database are initially associated with the SYSTEM tenant

– A SET TENANT statement must be issued to associate a connection with a user-defined tenant



User-defined tenants

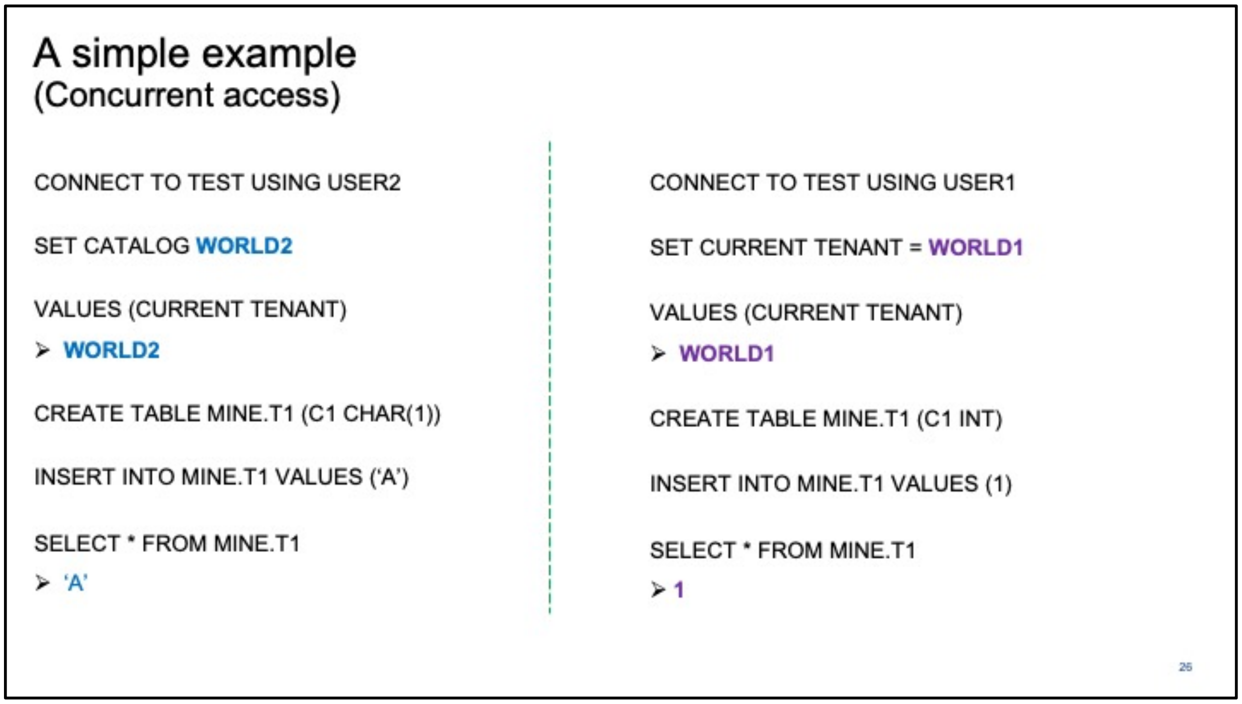
A DBADM can create a tenant to set up an independent catalog namespace within a Db2 database

All connections made to a database are initially associated with the SYSTEM tenant

– A SET TENANT statement must be issued to associate a connection with a user-defined tenant

A simple example (Setup)

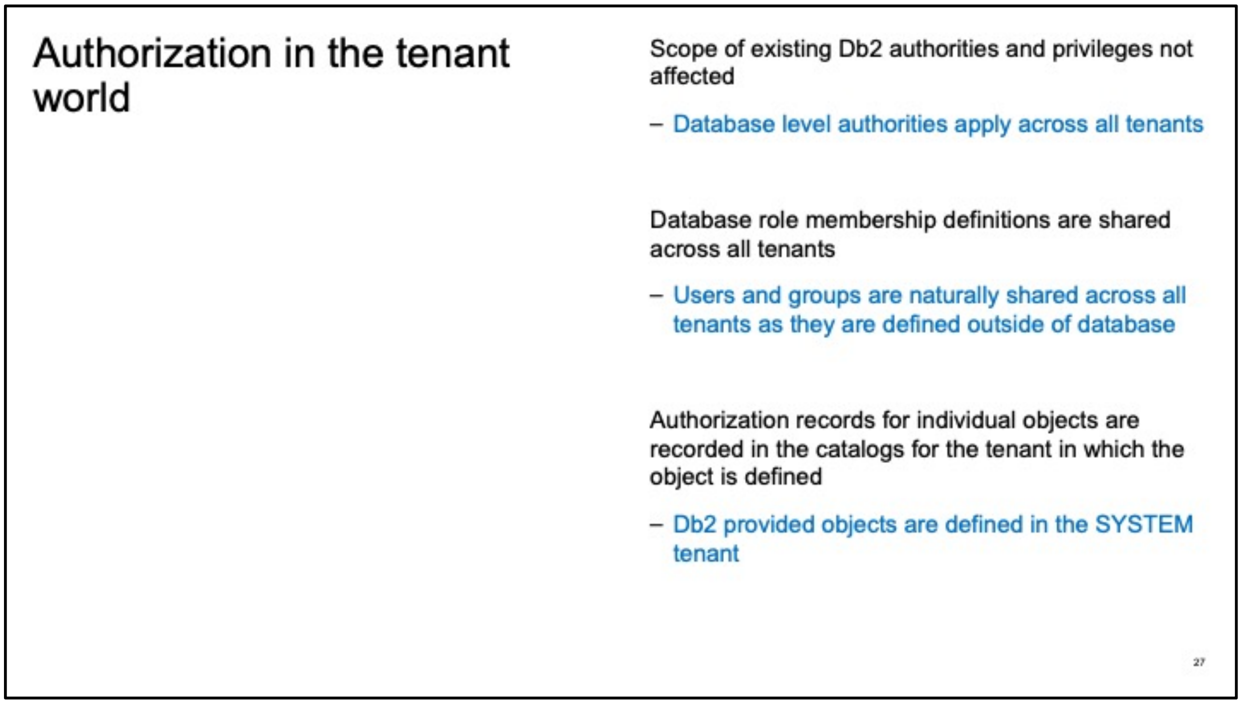
```
CREATE DATABASE TEST

CONNECT TO TEST

CREATE TENANT WORLD1
GRANT USAGE ON TENANT WORLD1 TO USER1

CREATE TENANT WORLD2
GRANT USAGE ON TENANT WORLD2 TO USER2
```



A simple example
(Concurrent access)

```
CONNECT TO TEST USING USER2

SET CATALOG WORLD2

VALUES (CURRENT TENANT)
➢ WORLD2

CREATE TABLE MINE.T1 (C1 CHAR(1))

INSERT INTO MINE.T1 VALUES ('A')

SELECT * FROM MINE.T1
➢ 'A'
```

```
CONNECT TO TEST USING USER1

SET CURRENT TENANT = WORLD1

VALUES (CURRENT TENANT)
➢ WORLD1

CREATE TABLE MINE.T1 (C1 INT)

INSERT INTO MINE.T1 VALUES (1)

SELECT * FROM MINE.T1
➢ 1
```



Authorization in the tenant world

Scope of existing Db2 authorities and privileges not affected

– Database level authorities apply across all tenants

Database role membership definitions are shared across all tenants

– Users and groups are naturally shared across all tenants as they are defined outside of database

Authorization records for individual objects are recorded in the catalogs for the tenant in which the object is defined

– Db2 provided objects are defined in the SYSTEM tenant

Authorization in the tenant world (An example)

A query executed while in tenant TENANTA refers to:

- A user-defined table, MYSCHEMA.T1, in that tenant
- The Db2 provided SYSPROC.MON_GET_CONNECTION table function

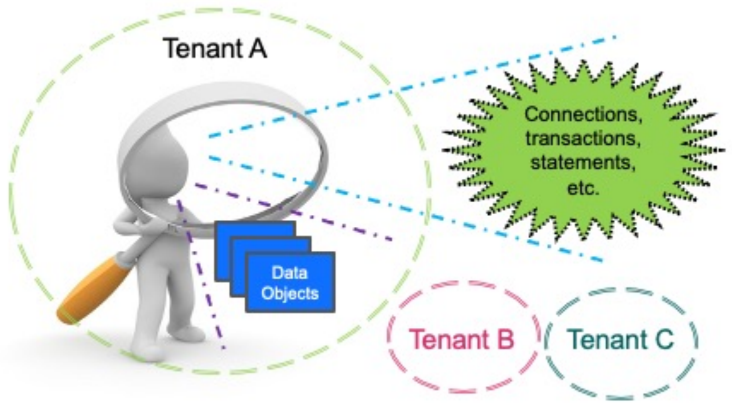
The authorization check for accessing the table MYSCHEMA.T1 is resolved using:

- The relevant database authorities
- The relevant authorization catalog tables within TENANTA

The authorization check for executing the Db2-defined table function SYSPROC.MON_GET_CONNECTION is resolved using:

- The relevant database authorities
- The relevant authorization catalog tables within the default SYSTEM tenant

Monitoring in the tenant world



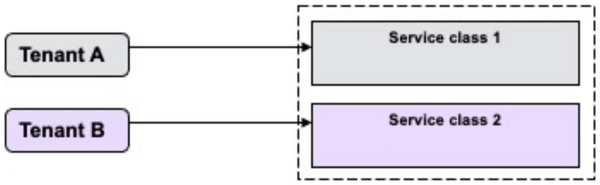
- Output from operation-centric table functions shows contributions from all tenants
- Output from object-centric table functions is restricted to rows related to the invoking tenant

Workload management in the tenant world

TENANT has been added as a new connection attribute for defining a Db2 workload

➢ This allows for tenant specific monitoring and control

- A workload defined for a specific tenant name will cause all connections associated with that tenant to map to that workload



Final words

When will we see tenant?

Namespace development is nearing completion

- Did not make Db2 11.5.6

➢ You can already see “tenant” details popping up in recent Db2 11.5 updates

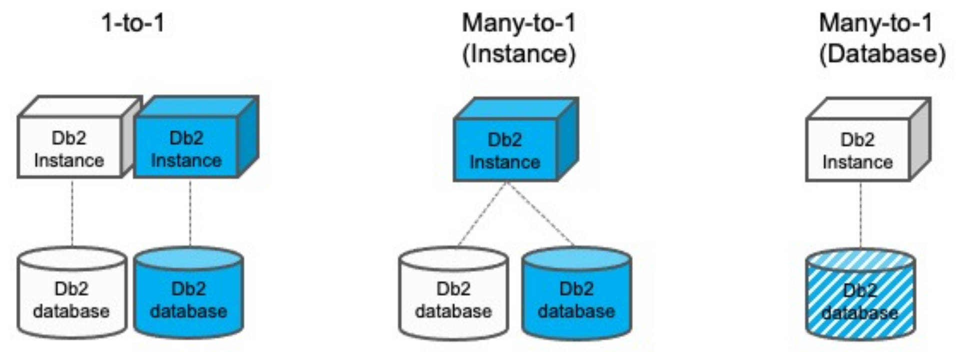
Deployment to the Db2 CSE family products may be staged

– Depending on delivery vehicle may be first exposed in Db2 Warehouse followed by other productss

– May be shipped in next Db2 version (i.e., not in Db2 11.5)

Expect update on planned delivery vehicle by 2H2021

Traditional Db2 database “multi-tenancy” configurations



The benefits of the tenant concept

Provides a lower-cost option for ”hosting” different sets of users/applications

Provides an easier path to testing within production environments

Makes migration from other database platforms easier

– Not forced to “flatten” database to schema unless it makes sense to do so

Why else might you consider using tenant?

Provides a way to apply WLM controls to workloads using specific data objects

Provides a way to sub-divide an existing database into independent zones

– Allows delegation of administration and security across subsets of database

Your feedback is welcome (and needed!)

What should be done next?

- Which tenant authorizations are needed first? Are there others that are valuable?

Many other possibilities exist such as

- Automatic assignment of user to tenant
- User-defined global objects

Need feedback from you to help guide future investment!

Questions?