Introduction to Multi-tenancy

Gus Bjorklund
October 2013

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Introduction

The session explores the upcoming inbuilt multi-tenancy capabilities included in the OpenEdge 11 RDBMS.

Learn how multi-tenant support impacts queries, indexes, sequences, and the physical storage of tenant data, as well as the operational activities that DBAs perform.

Please ask questions as we go

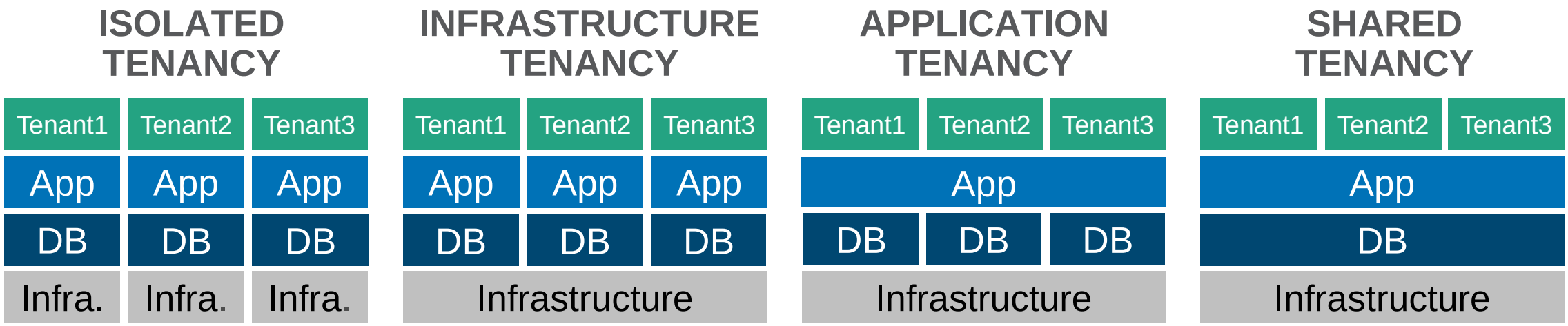
Sometimes I may not explain well enough, or perhaps you just want to know more

What is a Tenant Anyway?

Tenants are:

- Named groups of people (users) that are related in some (organizational) way, share data, and use the same application(s)
- They might work in the same company, work in same division or dept. of a larger company, or belong to the same club
- Tenants don't know others may be using the same system
- For example, tenants could be the makers of these fine refreshing beverages:

Multi-tenancy Options Continuum



Isolating ***Sharing***

- Easier customization, security
- Simpler throttling control
- Target dissimilar customers
- No transformation

- Better economy of scale
- Simpler management
- Target like-customers
- Least cost to serve

Why Multi-tenancy? Vendors Want to ...

- Increase infrastructure efficiency
  - Do the job will less hardware or more with same
- Reduce operational and administrative labor
  - Do the job with less work
- Decrease operating costs
  - Allow higher profits to provider
  - Allow lower prices to customers

SaaS Application Customers Want

- Low startup cost
- Fast deployment
- 100% uptime
- Responsive applications
- Data security (well, they *should* anyway)
- Low prices

Why *Database* Multi-tenancy?

- Lower SaaS application development cost and time
- Lower SaaS application deployment cost and time
- Lower operational costs
- Lower administrative costs
- Provide more flexibility for OpenEdge ISV partners
- Provide more flexibility for OpenEdge customers

In 10.2B, you can do this:

Extra “Tenant ID” Column For Multitenancy

Tenant ID	Cust ID	Name
A	1	Lift Line Skiing
A	2	Urban Frisbee
A	3	Hoops Croquet
B	1	Fanatical Athletes
B	8	Game Set Match
B	9	Lift Line Skiing
C	2	High Tide Sailing
C	7	Pedal Power
C	9	Hoops Croquet

FOR EACH CUSTOMER WHERE (TenantID = A) and (regular stuff):

What's wrong with that?

Do we need more?

PROGRESS

It Works, But There Are Just a Few Small Disadvantages

- Invasive: you have to change a lot of 4GL code
- Mistakes likely – then data given to wrong tenant
- Lock conflicts can occur among tenants
- Suboptimal performance
  - Low locality of reference
  - Low database buffer cache efficiency
  - Low I/O efficiency

And Still Other Disadvantages

- Per tenant bulk operations difficult
  - Backup, restore, reindex, delete, copy, move
- Tenant-level performance analysis difficult
- Tenant resource consumption metrics difficult
- Tenant resource utilization controls difficult
- And a bunch of other things

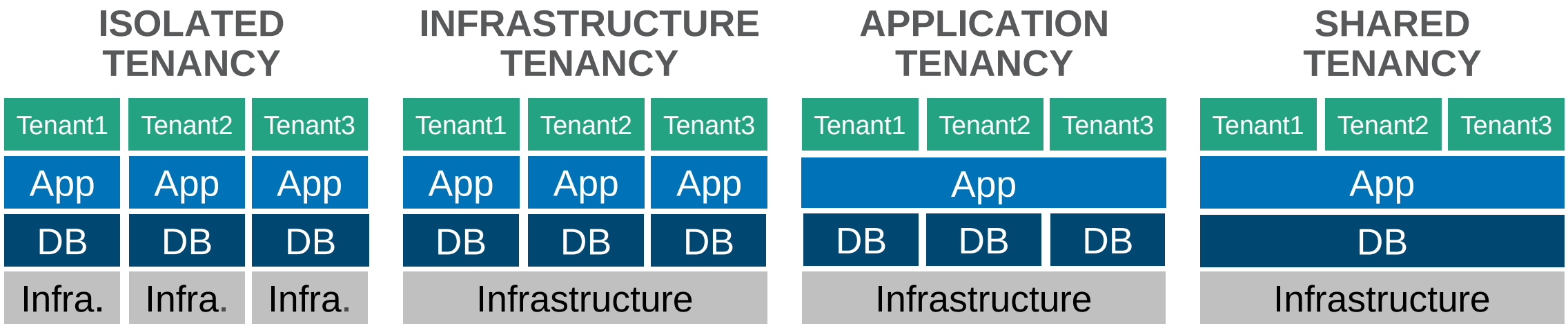
Yes! You do need more.
And with OpenEdge 11, you get more.

The RDBMS has inbuilt multi-tenancy for both 4GL and SQL applications

Main purpose of OpenEdge 11 inbuilt multi-tenancy is to: Reduce costs for SaaS vendors

How does it work?

Multi-tenancy Options Continuum

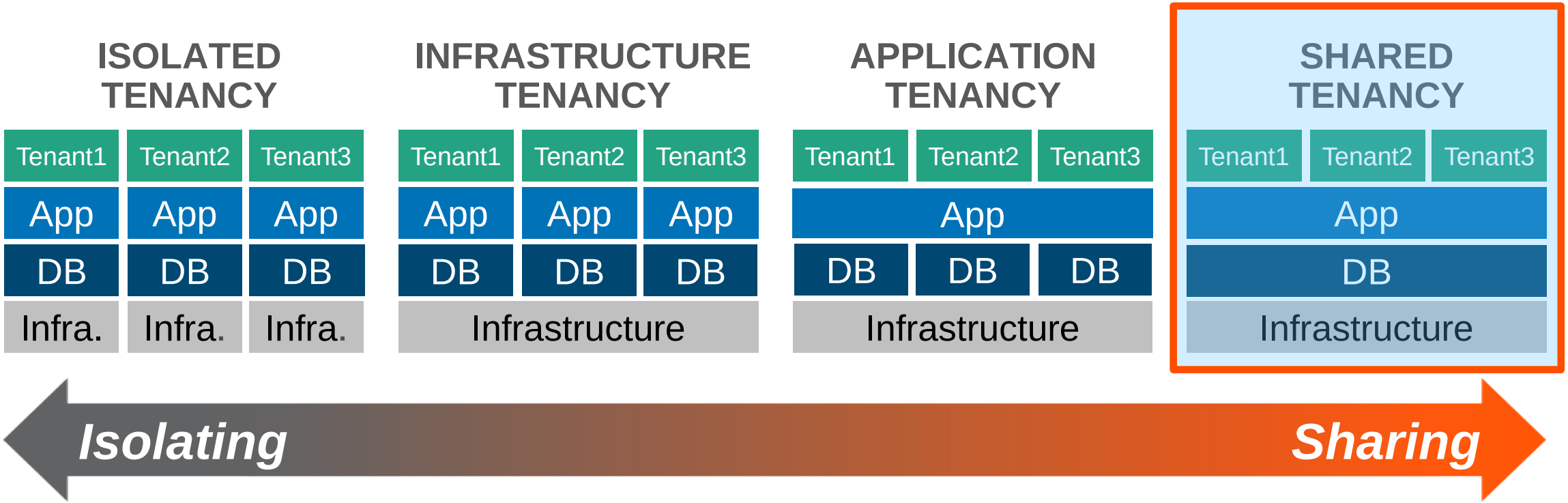


Isolating ***Sharing***

- Easier customization, security
- Simpler throttling control
- Target dissimilar customers
- No transformation

- Better economy of scale
- Simpler management
- Target like-customers
- Least cost to serve

Multi-tenancy Options Continuum



- Easier customization, security
- Simpler throttling control
- Target dissimilar customers
- No transformation

- Better economy of scale
- Simpler management
- Target like-customers
- Least cost to serve

OpenEdge Multi-tenant Tables: NO Extra Column for Tenant ID

OpenEdge Multi-tenant Tables: NO Extra Column for Tenant ID

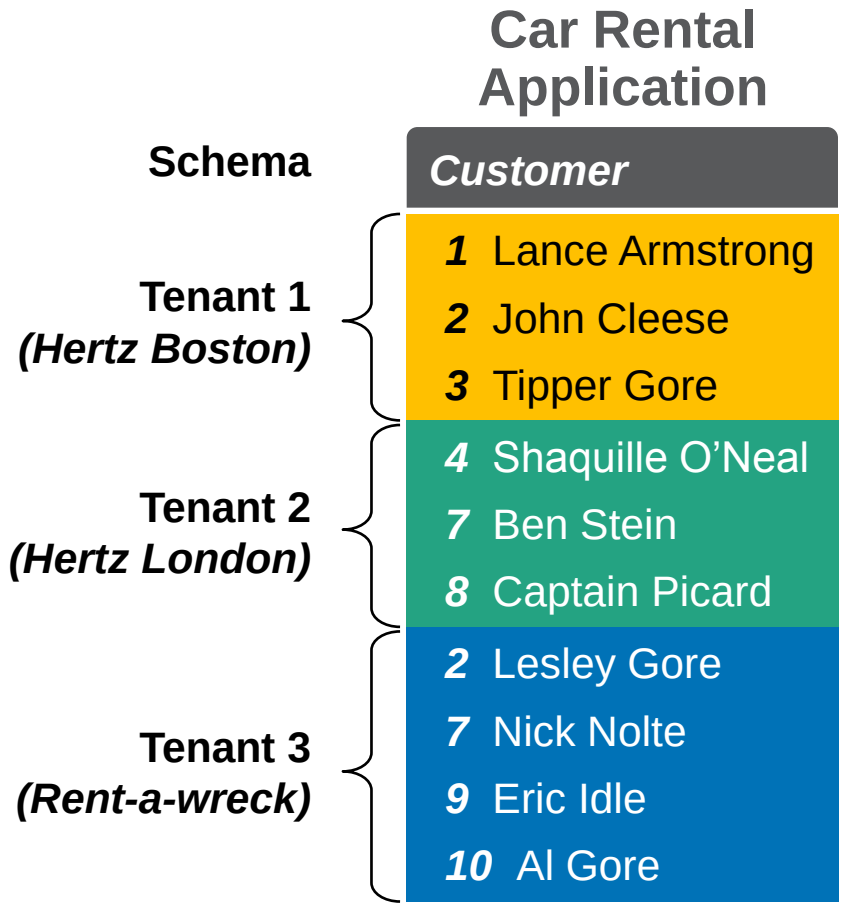
Cust ID	Name
1	Lift Line Skiing
2	Urban Frisbee
3	Hoops Croquet
1	Fanatical Athletes
8	Game Set Match
9	Lift Line Skiing
2	High Tide Sailing
7	Pedal Power
9	Hoops Croquet

FOR EACH CUSTOMER:

OE 11 Multi-tenant Tables

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Multi-tenancy built into the database
- Data physically partitioned by tenant identity
- Tenants share same schema definition
- **Minimal** application changes
  - Just set a per-database tenant name

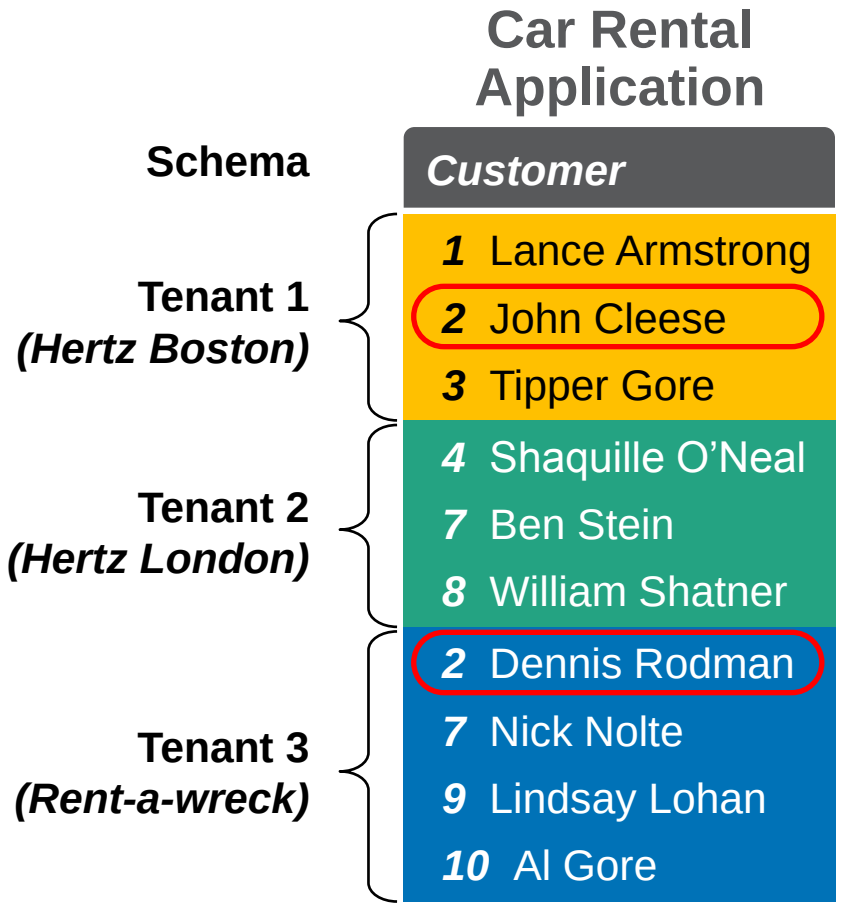


**Fictitious example*

Multi-tenant Tables: Data Access

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Keys unique per tenant partition

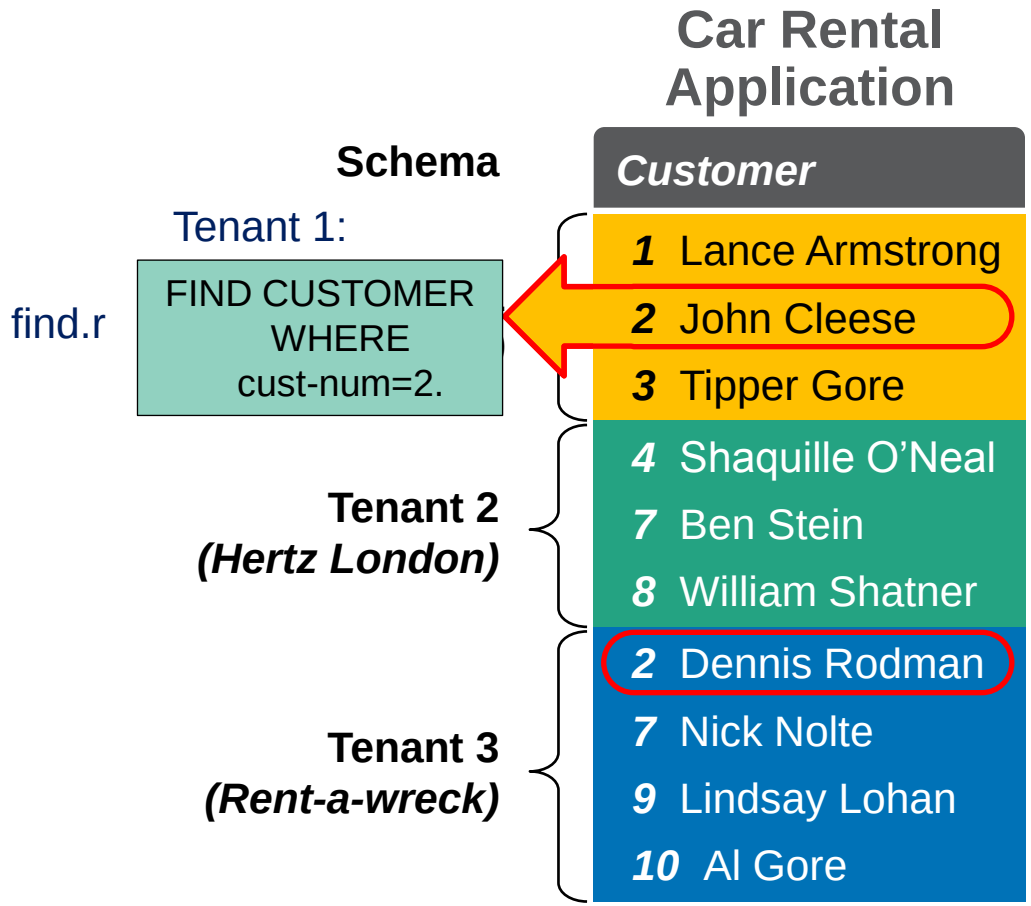


**Fictitious example*

Multi-tenant Tables: Data Access

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Keys unique per tenant partition
- Query is tenant-specific
  - Authenticate as tenant
    - _User
    - Client Principal
  - Assert tenant identity

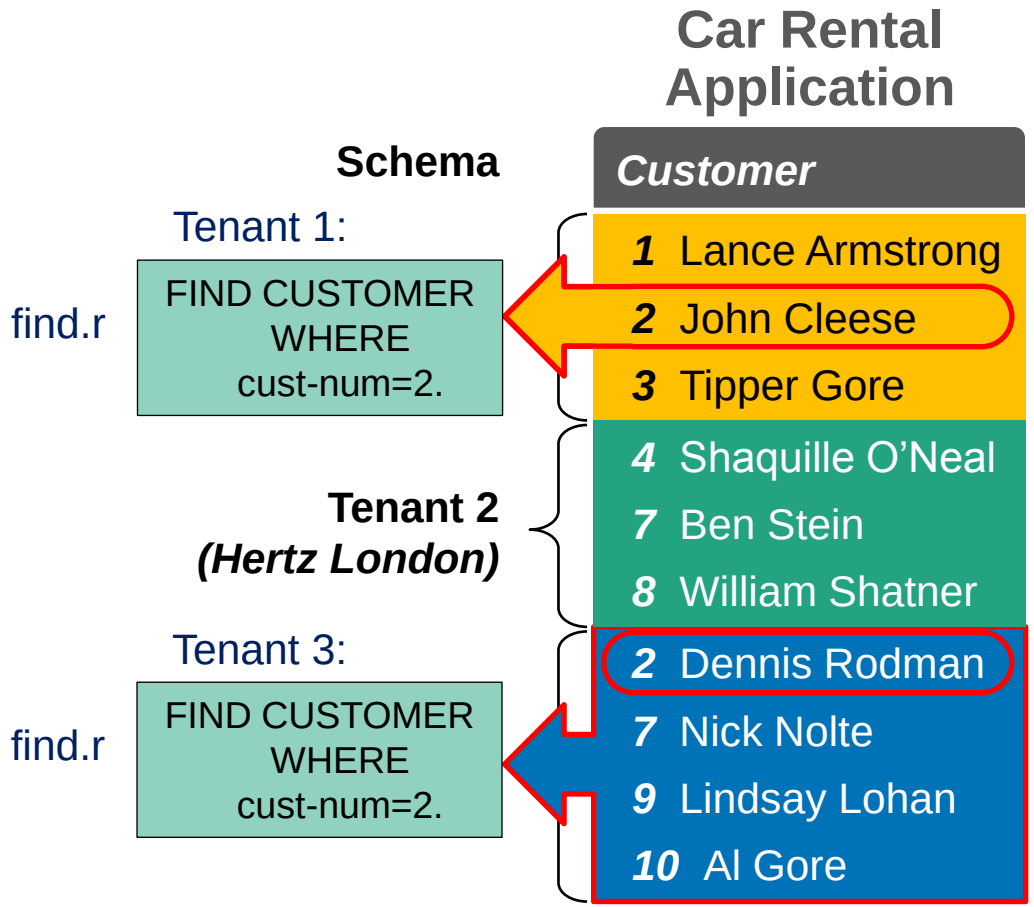


**Fictitious example*

Multi-tenant Tables: Data Access

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Keys unique per tenant partition
- Query is tenant-specific
  - Authenticate as tenant
    - _User
    - Client Principal
  - Assert tenant identity



**Fictitious example*

Multi-tenant Tables: Data Access

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Keys unique per tenant partition
- Query is tenant-specific
- "Super-tenant" query
  - Authenticate & assert identity
  - No data of their "own"
  - Access to all tenant data by tenant ID or name

Schema

Super-tenant:

FOR EACH customer
TENANT-WHERE Tenant-id > 0:
DISPLAY cust-num, name.

Car Rental Application

Customer
1 Lance Armstrong
2 John Cleese
3 Tipper Gore
4 Shaquille O'Neal
7 Ben Stein
8 William Shatner
2 Dennis Rodman
7 Nick Nolte
9 Lindsay Lohan
10 Al Gore

**Fictitious example*

Multi-tenant Tables: Data Access

Multi-tenancy **Simplifies Development of Multi-tenant Applications**

- Keys unique per tenant partition
- Query is tenant specific
- “Super-tenant" query
- Row-level tenant identification
- Virtual column available for display or selection (not in table definition)

Super-tenant:

FOR EACH customer
TENANT-WHERE
Tenant-id > 0:
DISPLAY
BUFFER-TENANT-ID(cust),
cust-num, name.

Schema

Car Rental Application

Customer
1 1 Lance Armstrong
1 2 John Cleese
1 3 Tipper Gore
2 4 Shaquille O’Neal
2 7 Ben Stein
2 8 William Shatner
3 2 Dennis Rodman
3 7 Nick Nolte
3 9 Lindsay Lohan
3 10 Al Gore

**Fictitious example*

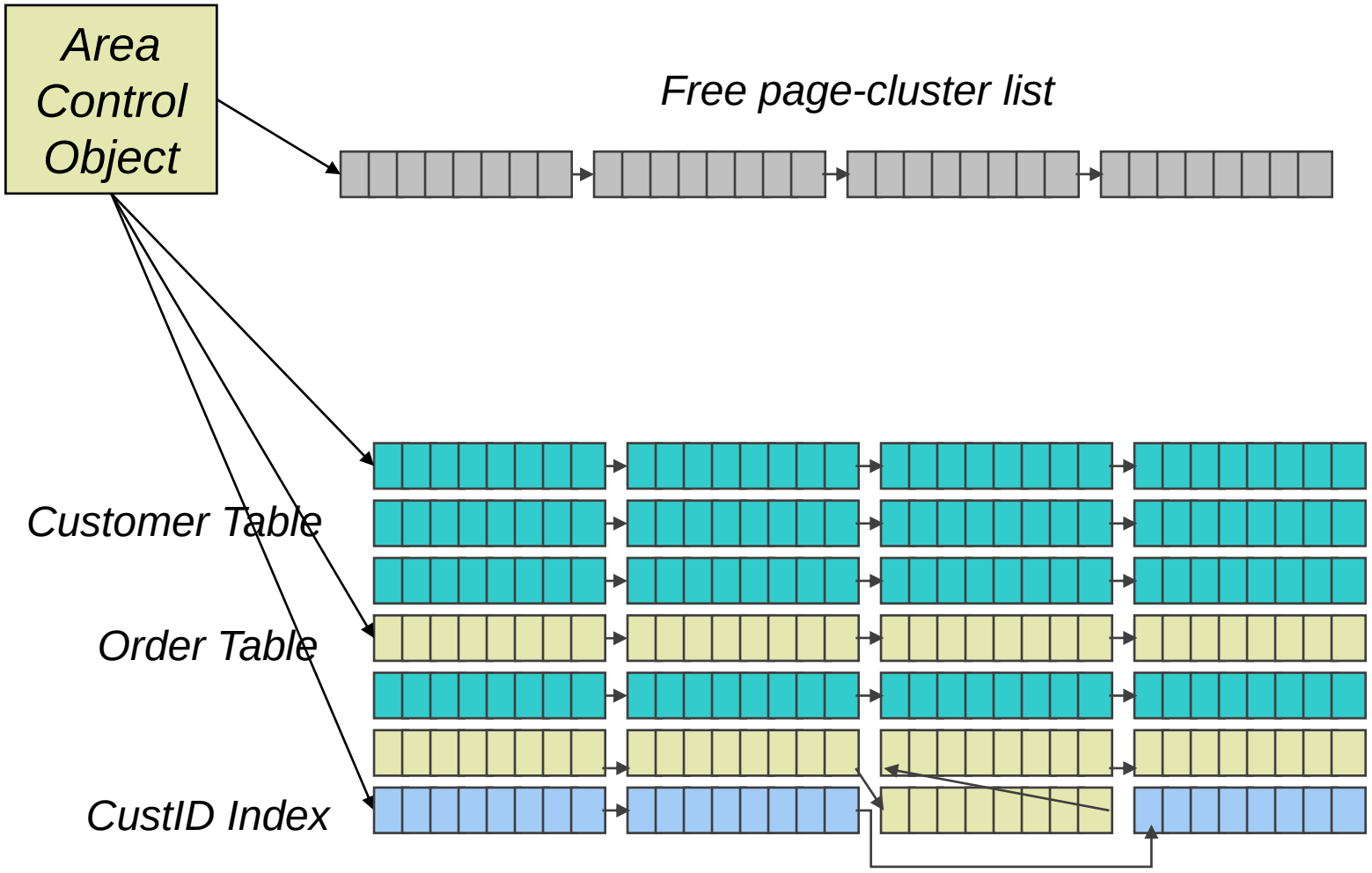
PROGRESS

3 Types of Tenants

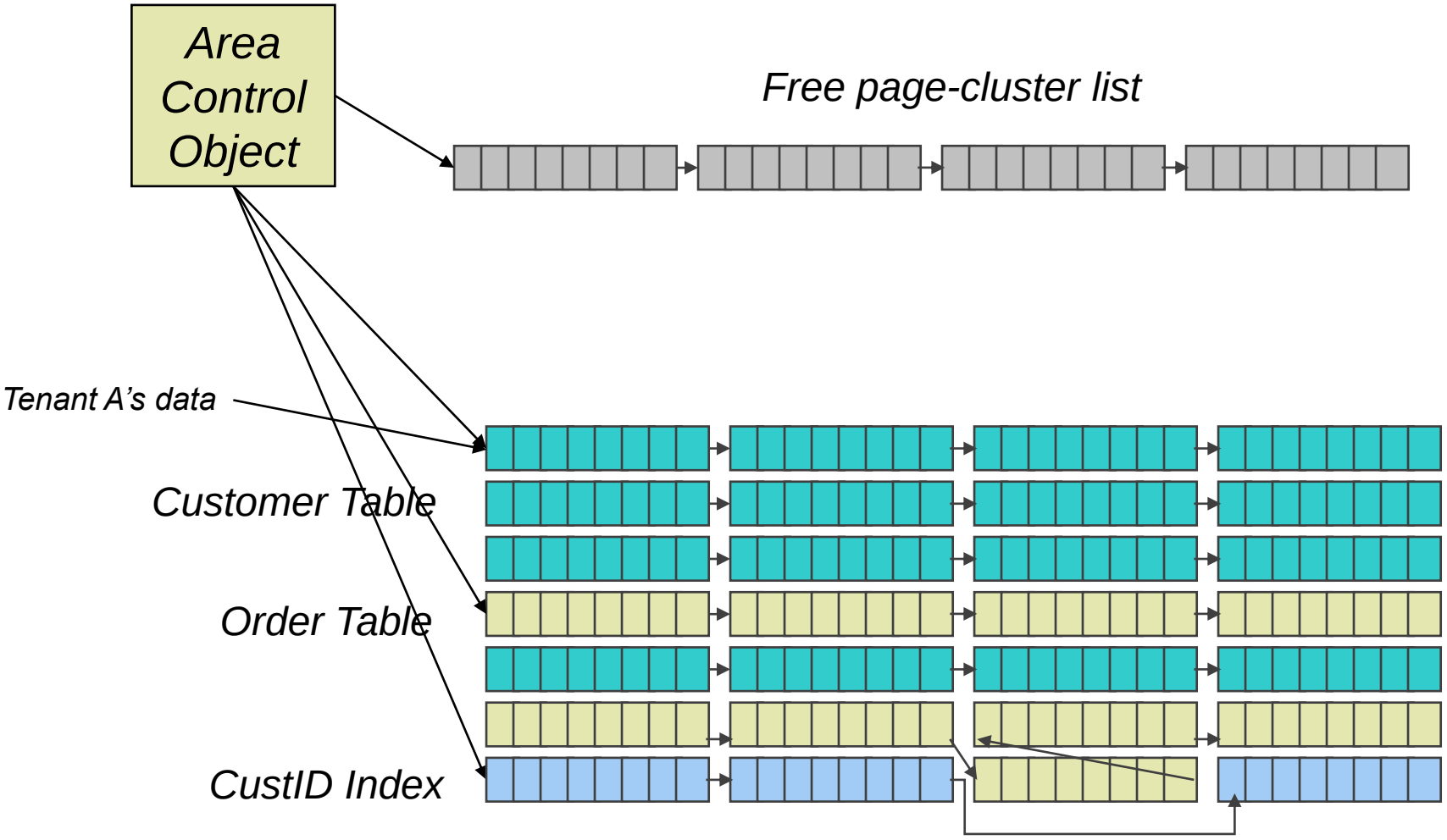
- Default
- Regular
- Super

Tenant Data Storage

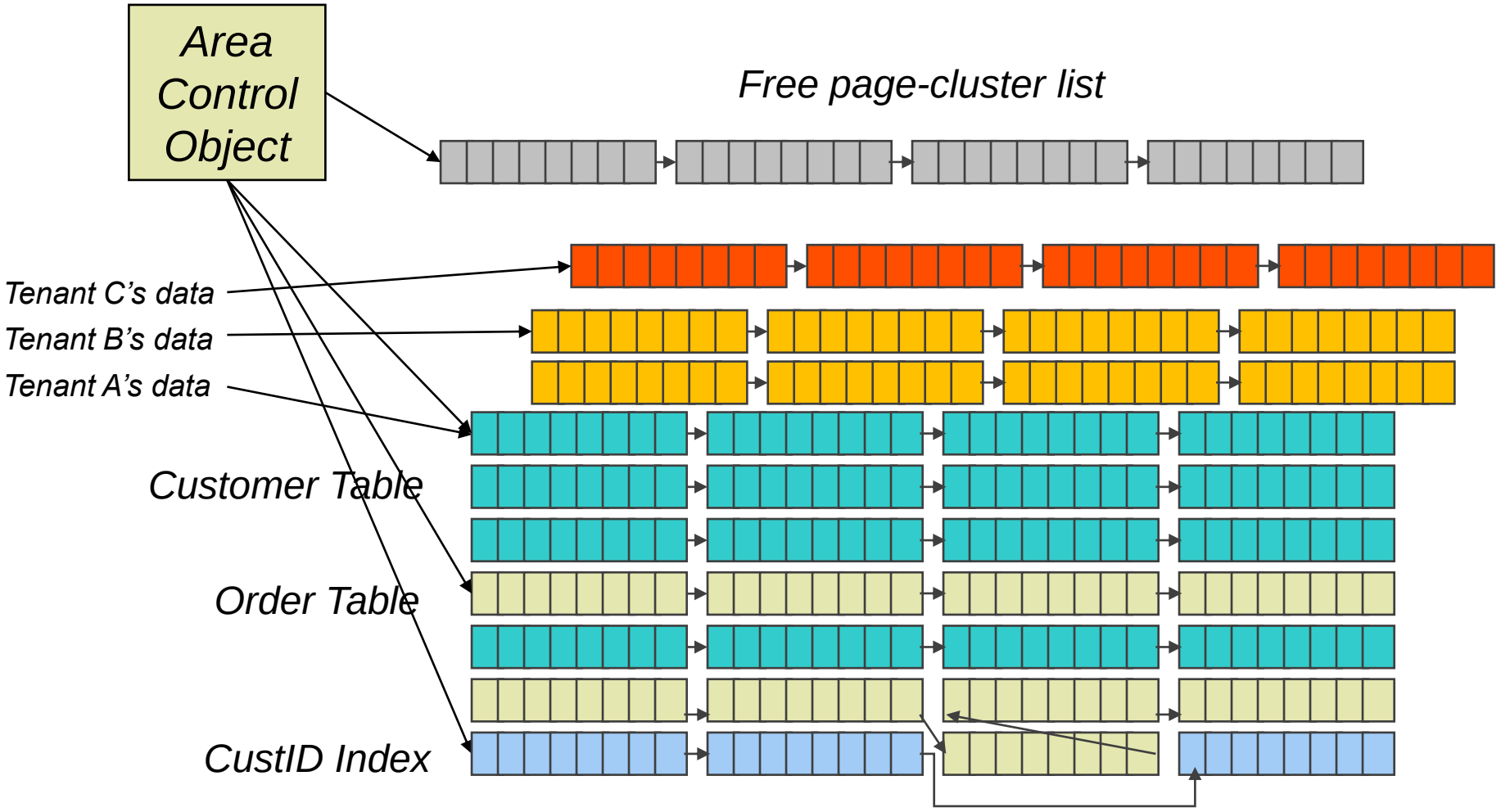
Multitenant Storage Area Structure: Tenant Data Partitions



Multitenant Storage Area Structure: Tenant Data Partitions

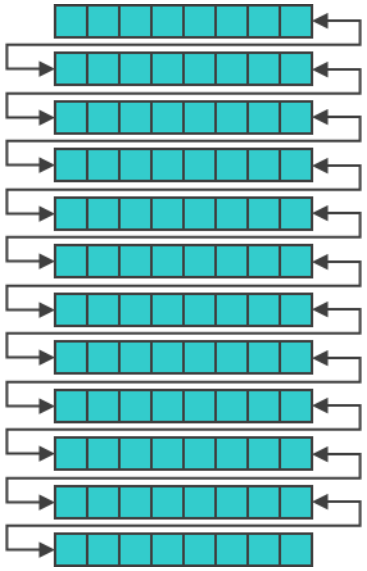


Multitenant Storage Area Structure: Tenant Data Partitions



Tables: Physical Storage View (Type ii Data Areas)

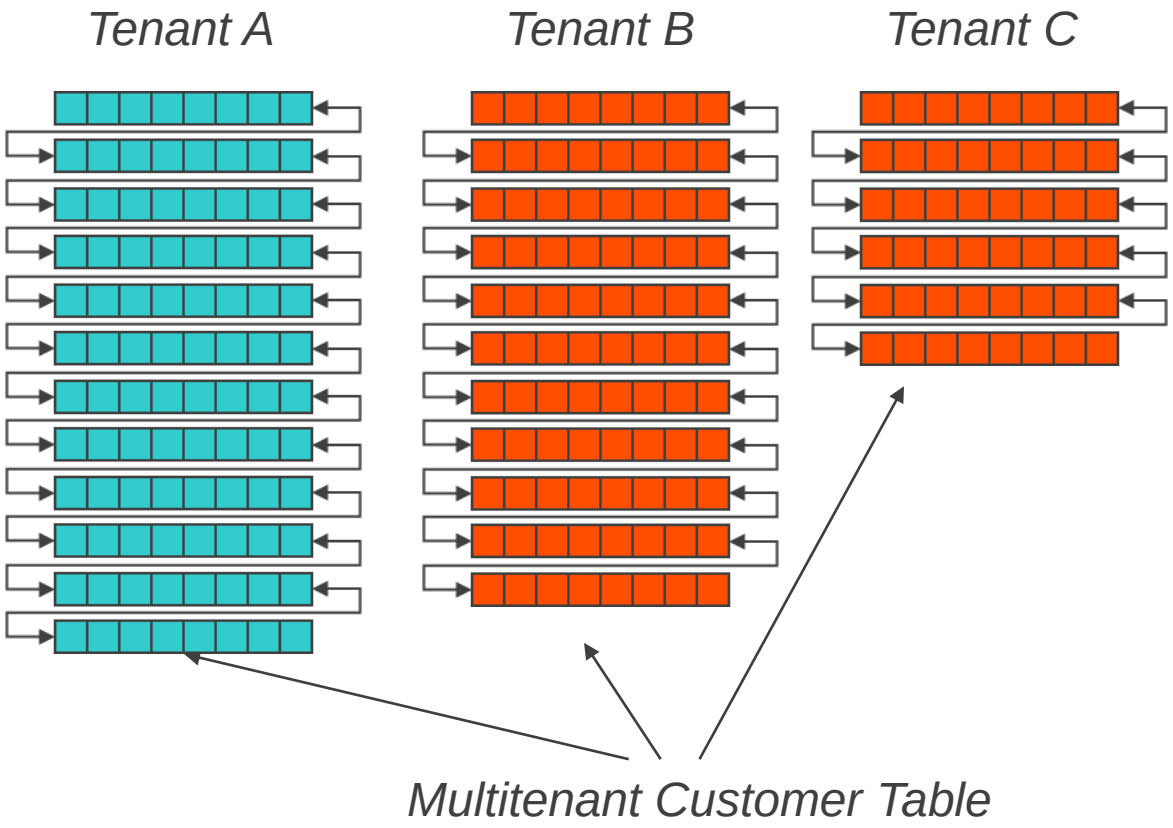
Linked list of page-clusters



Shared Customer Table

OpenEdge Multitenant Tables: Automatic Table Partition for Each Tenant

Linked list of page-clusters for each tenant's data



Numbers

500 tables

10 indexes per table (maybe a bit high)

100 tenants

= (500 * 100) + (500 * 10 * 100)

= 505,000 partitions !!!

Strategies for Storage Layout

With very many partitions, you have to keep it simple.

Strategies for Storage Layout

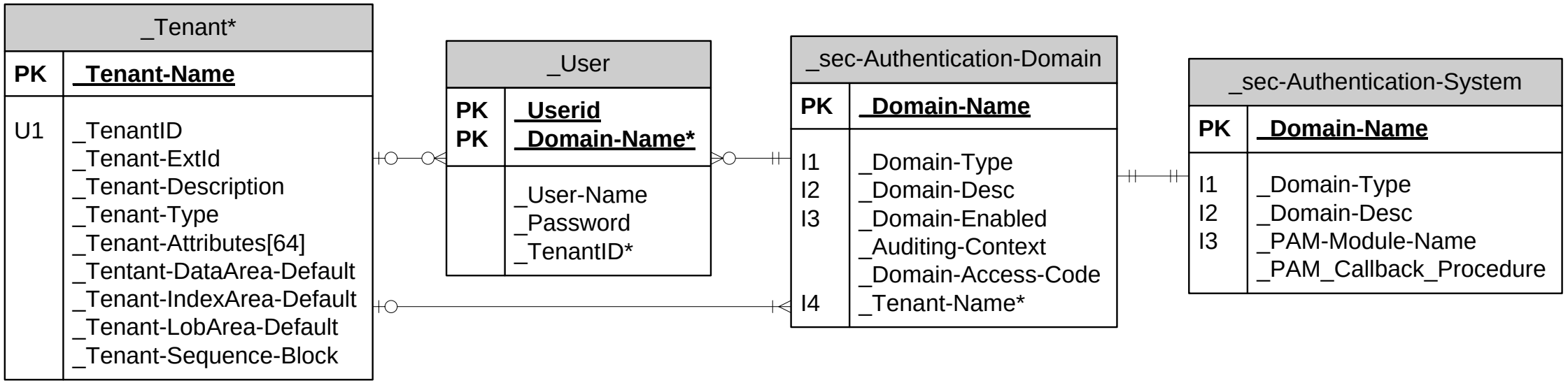
- Shared tables all in one area
- All tenants in one area
- 5 tenants per area
- "stripe" p partitions over n areas (p >> n)
- One storage area per tenant
- 3 areas per tenant (data, index, lob)

Tenants have their own data partitions

How does database know to which tenant a user belongs ?

DOMAINS

- A tenant is a collection of users
- A user is a "person"
- A *security domain* is named set of rules ("policies") for how a group of users identity and tenant association is verified
- Every tenant must have *at least one* domain



DOMAINS

- When you create a tenant, you must also create a domain.
- The domain specifies how user identity is validated
- Possibilities include:
  - _user table has user name and password
  - operating system identity
  - external system like LDAP, Active Directory, etc.
  - Your 4GL code

How Users and Tenants Are Identified

- Users have names
- Tenants have domains
- Domains have names
- Together the two names are unique

user-name@domain-name

DOMAINS

When you log in
you must specify user ID and
you must also specify a domain.

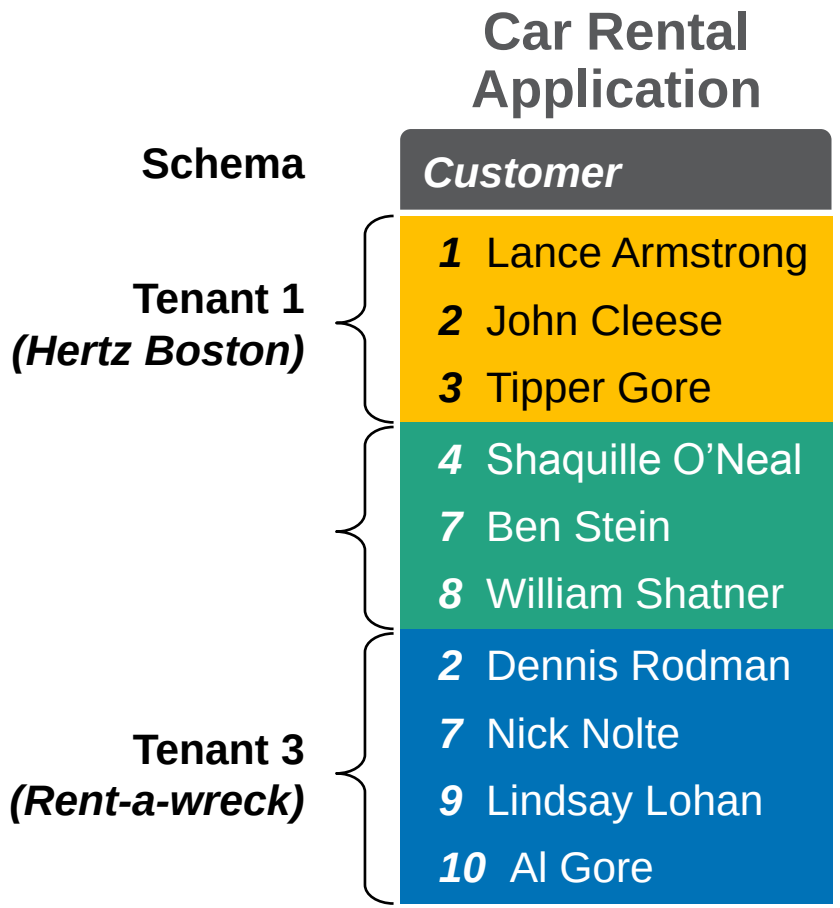
for example:
mpro –db foo –U user@domain –P password

we will see some other ways later.

Multi-tenancy: Data Access, Sharing

Tenant Groups

- Some tenants can share the same data/partition
- Employee access to shared customer list

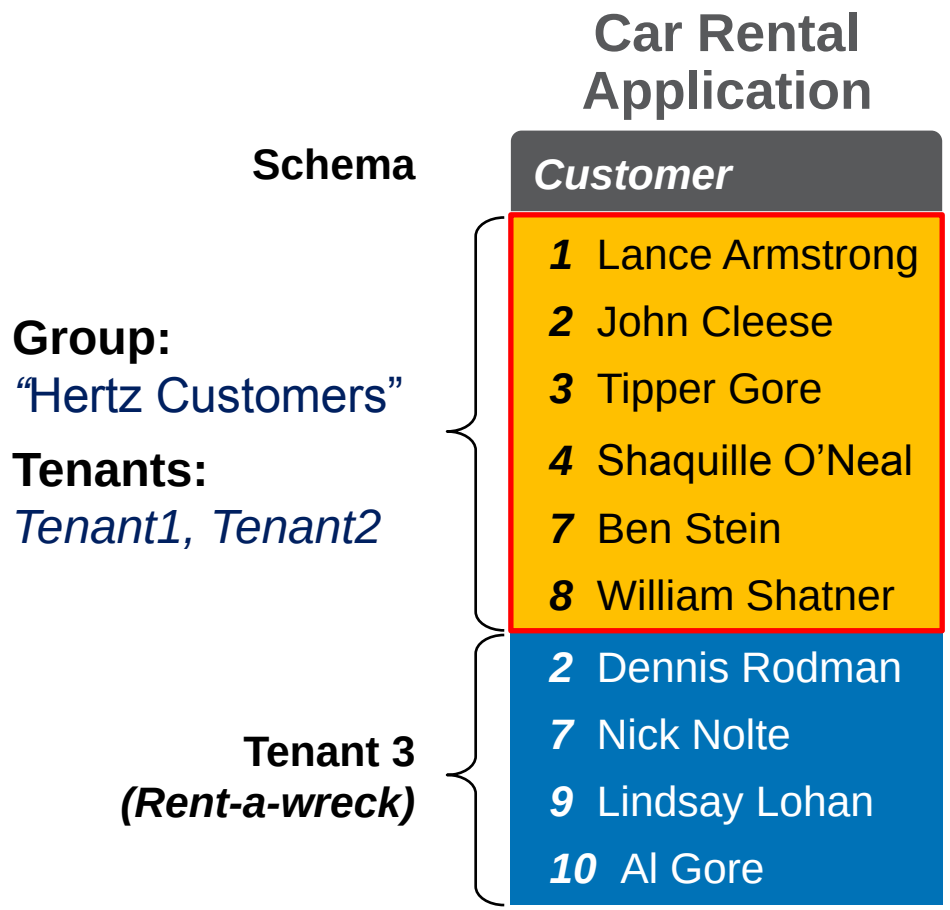


*Fictitious example

Multi-tenancy: Data Access, Sharing

Tenant Groups

- Some tenants can share the same data/partition
  - Employee access to shared customer list

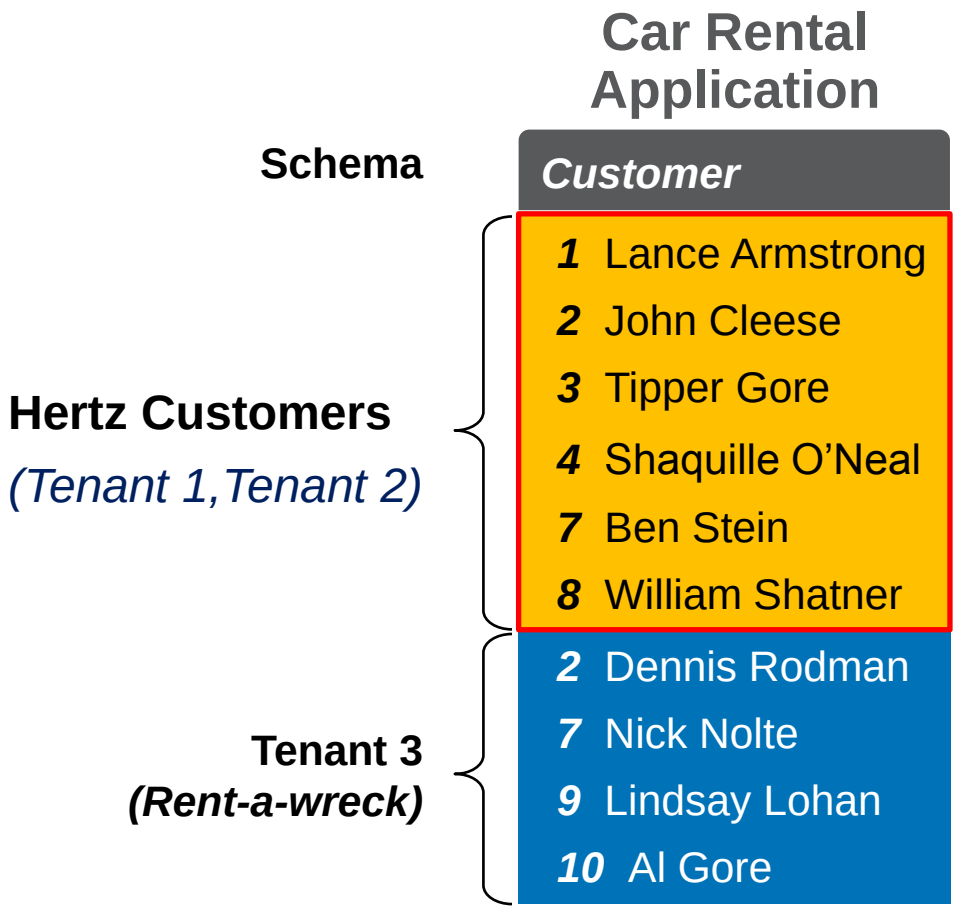


*Fictitious example

Multi-tenancy: Data Access, Sharing

Tenant Groups

- Some tenants can share the same data/partition
  - Employee access to shared customer list
- Data exists for the life of the group
  - e.g. Regional data
- Row identity associated with group
  - BUFFER-GROUP-ID()
  - BUFFER-GROUP-NAME()
- Group membership is per table

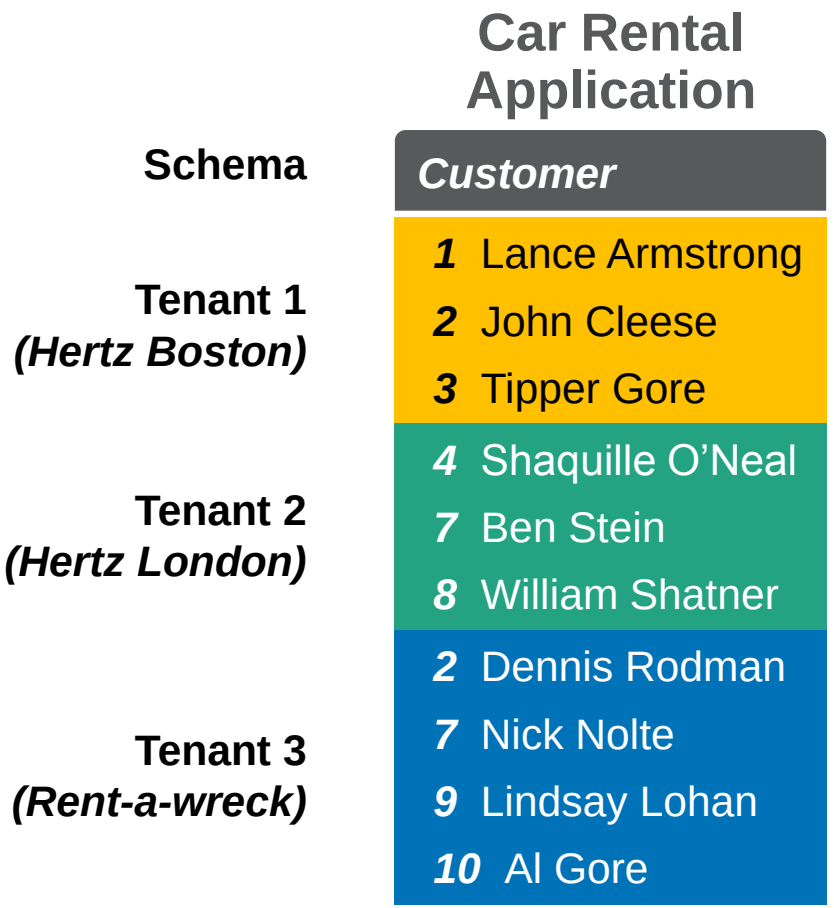


*Fictitious example

Multi-tenancy: Data Model

Tenant Groups

- Multi-tenant objects
  - Tables and associated indexes & LOBs
  - Sequences
- Shared objects still available
  - Same as today
- Shared only, not multi-tenant
  - Triggers & stored procedures
  - Initial values
- Limits
  - Support for up to 32,767 tenants

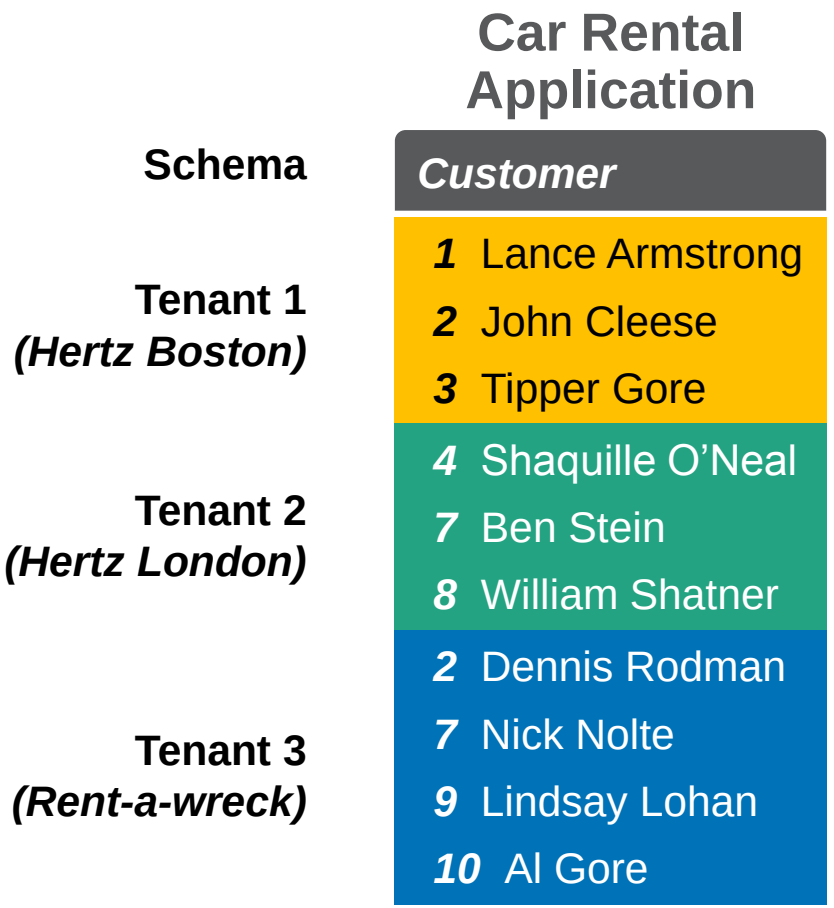


*Fictitious example

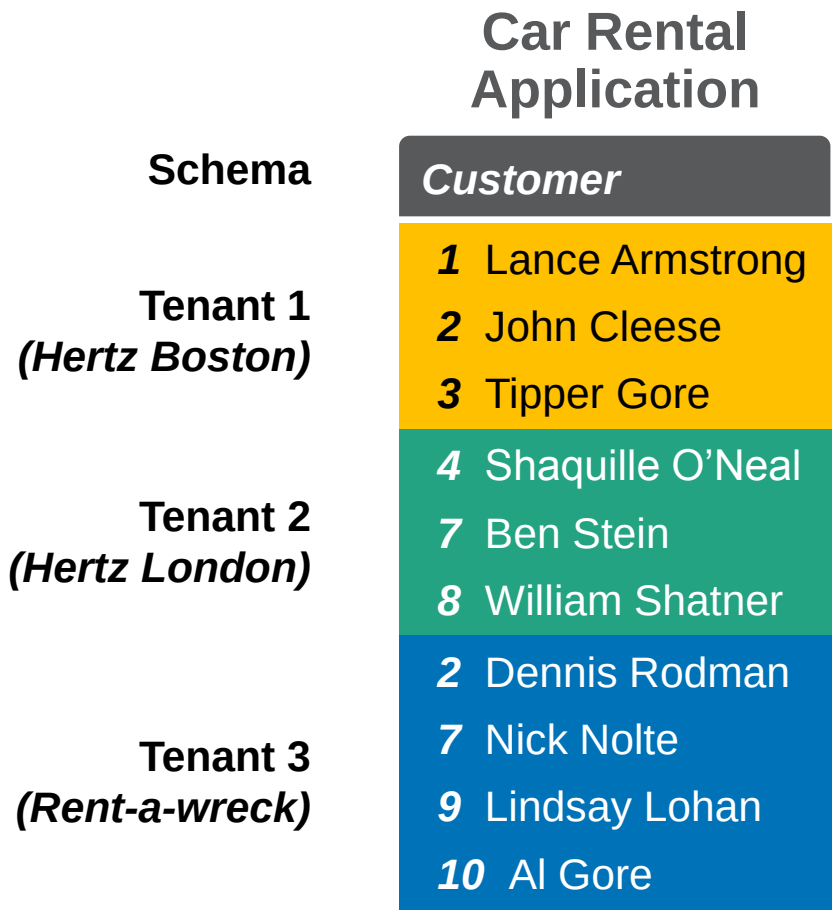
Multi-tenancy: Tenant Provisioning

Managing Tenants

- Tenant creation: ABL, APIs, DDL & GUI
  - Programmatic tenant provisioning
  - Tenant partition creation optional
  - Tenant level activation/deactivation
- Identification (via "_Tenant" table)
  - Database specific tenant ID
  - User friendly name: "Hertz, Boston"
  - App specific ID (could be UUID)
- Resource access
  - Runtime security by user by tenant
  - Governors: Limit resource usage



Multi-tenant Tables: Operational Features



Operational Features

- Tenant partition maintenance
  - Tenant-specific object move
  - Add/drop tenants/objects
  - Data dump/load
  - .df support
  - Index maintenance tools
- Monitoring
  - Promon, VSTs
  - Analysis tools
  - .lg file (other log files)

Multi-tenant Identity

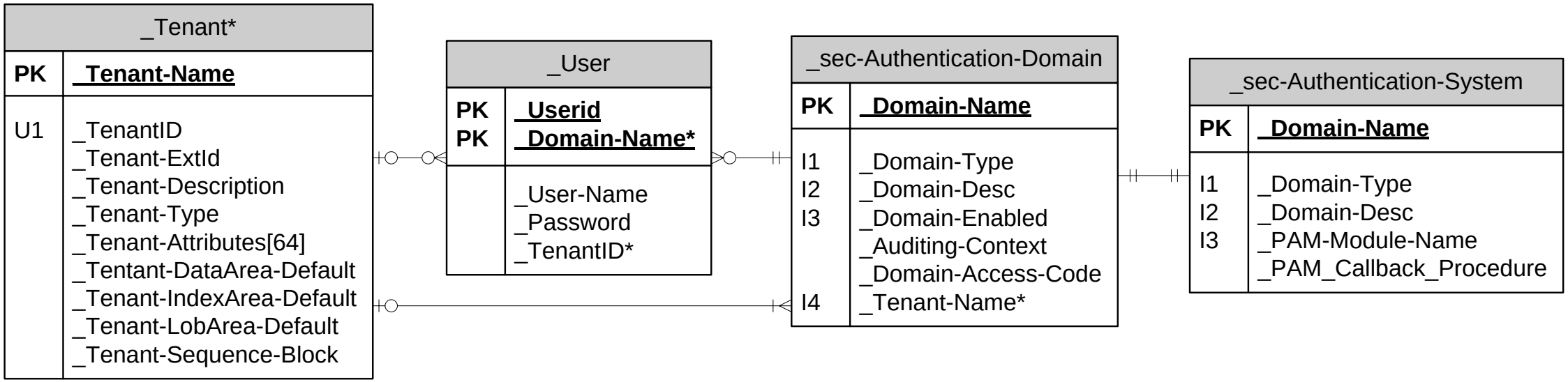
Who am I? Why am I here?

The _User table (ABL & SQL)

Assert identity via userId @ domainName

SETUSERID("**jsmith@hertz2**", "pwd").

CONNECT –U **jsmith@hertz2** –P pwd …



Multi-tenant Identity

Who am I? Why am I here?

- Using the Client Principal

```
/* Define & Create a client principal */
    DEFINE VAR hCP AS HANDLE.
    CREATE Client-Principal hCP.
/* Set, authenticate & seal – Nice new simple syntax in 11.0 */
    hCP:Initialize("jsmith@hertz2", sid, expr, "domain-access-code").
/* Validate, assert identity and use: */
    /* Establish DB connection Id */
    SET-DB-CLIENT(hCP).
    /* OR: Establish DB connection and session Id */
    SECURITY-POLICY:SET-CLIENT(hCP).
```

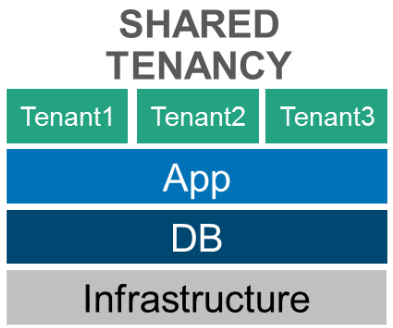
What data will you see ?
Depends who you are.
Database uses your identity to decide.

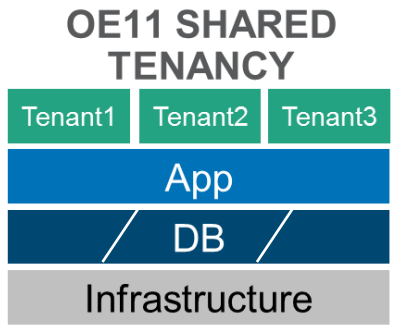
CLIENT-PRINCIPAL is basis for identity.

Switching identity with CLIENT-PRINCIPALs

```
SET-DB-CLIENT(hCP1).
/* now we are Alice */
FIND Customer WHERE name = "Alices Customer".
SECURITY-POLICY:SET-CLIENT (hCP2).
/* Now we are Bob */
CREATE Customer.
name = "Bobs Customer".
```

Multi-tenant Tables: Data Migration with DIY Tenant ID Column





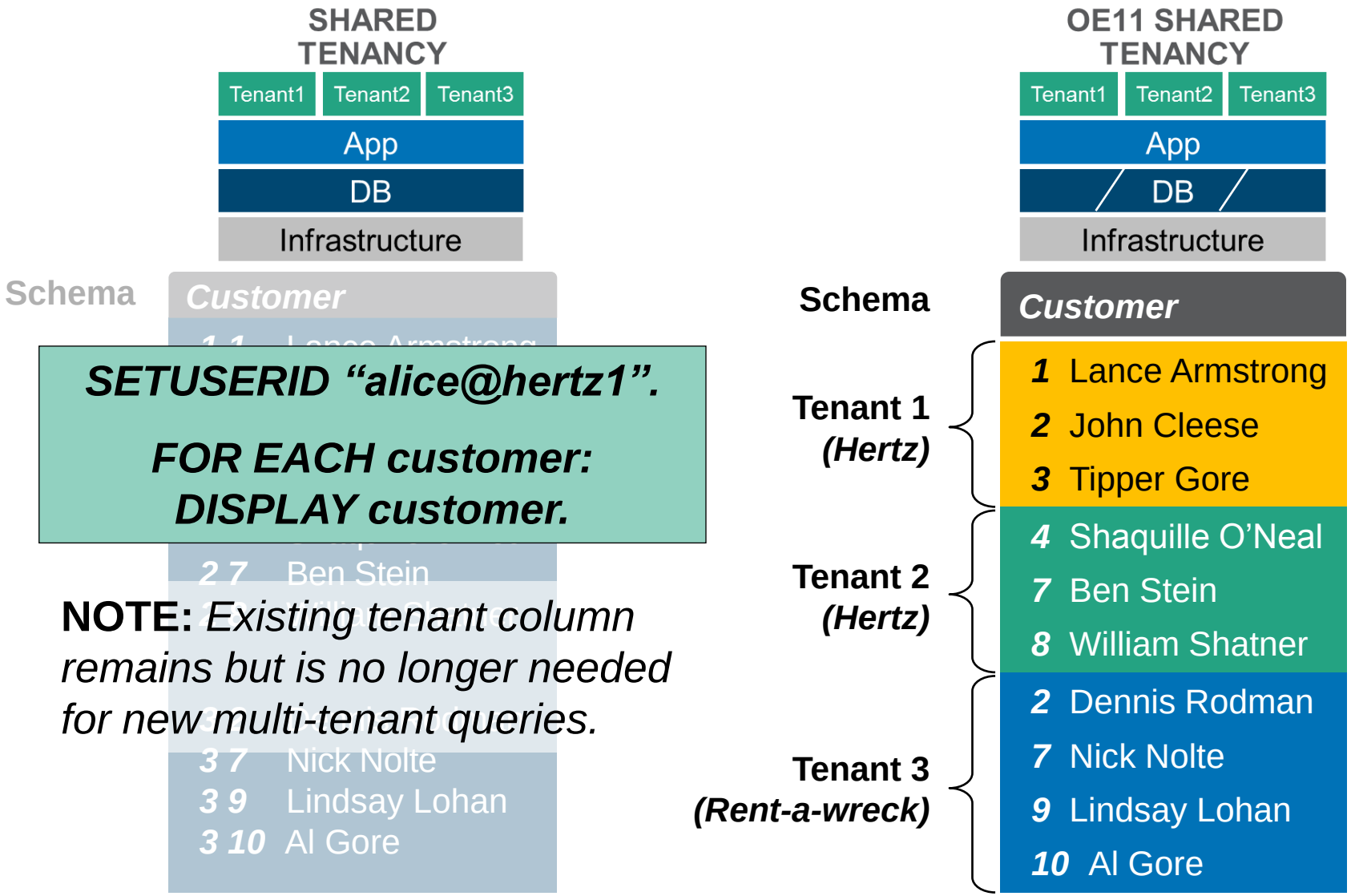
Schema

Default Partition

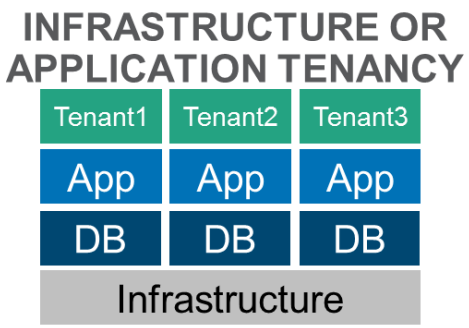
Customer	
1 1	Lance Armstrong
1 2	John Cleese
1 3	Tipper Gore
2 4	Shaquille O'Neal
2 7	Ben Stein
2 8	William Shatner
3 2	Dennis Rodman
3 7	Nick Nolte
3 9	Lindsay Lohan
3 10	Al Gore

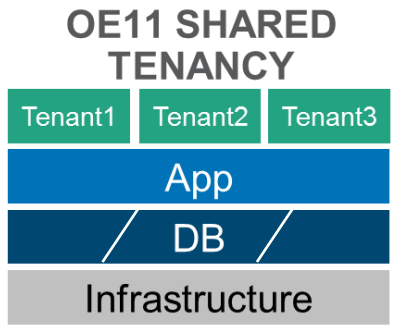
- Enable multi-tenancy on existing db
- Mark existing table as multi-tenant table
- Data in default tenant partition
- Set super-tenant identity
- Move data
- Truncate empty partition

Multi-tenant Tables: Data Migration with DIY Tenant ID Column



Multi-tenant Tables: Data Migration with Database per Tenant





DB #1
(Hertz Boston)

Customer
1 Lance Armstrong
2 John Cleese
3 Tipper Gore

DB #2
(Hertz London)

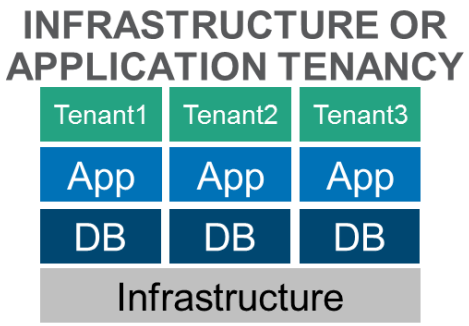
Customer
4 Shaquille O'Neal
7 Ben Stein
8 William Shatner

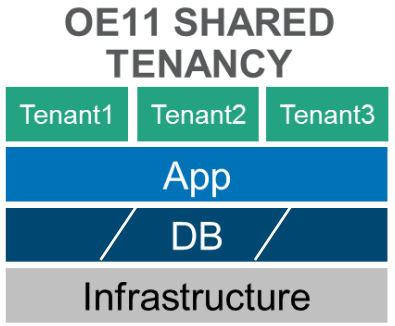
DB #3
(R.W.)

Customer
2 Dennis Rodman
7 Nick Nolte
9 Lindsay Lohan
10 Al Gore

PROGRESS

Multi-tenant Tables: Data Migration with Database per Tenant





DB #1 ***(Hertz Boston)***

Customer
1 Lance Armstrong
2 John Cleese
3 Tipper Gore

DB #2 ***(Hertz London)***

Customer
4 Shaquille O'Neal
7 Ben Stein
8 William Shatner

DB #3 ***(R.W.)***

Customer
2 Dennis Rodman
7 Nick Nolte
9 Lindsay Lohan
10 Al Gore

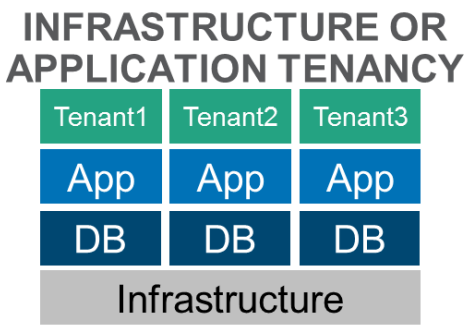
- Create **new** multi-tenant db
  - Can convert an existing one
  - Add tenants
  - Load multi-tenant schema
- Dump from current

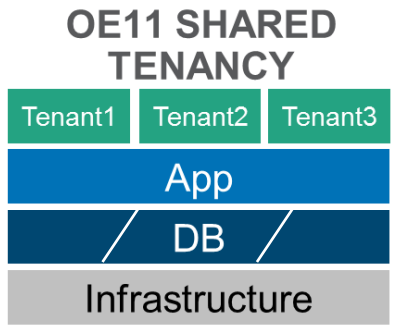
```
proutil  DB1 –C dump customer
```

- Load to new

```
proutil  MTdb –C load customer tenant  hertz2
```

Multi-tenant Tables: Data Migration with Database per Tenant





DB #1 *(Hertz)*

Customer
1 Lance Armstrong
2 John Cleese
3 Tipper Gore

DB #2 *(Hertz)*

Customer
4 Shaquille O'Neal
7 Ben Stein
8 William Shatner

DB #3 *(R.W.)*

Customer
2 Dennis Rodman
7 Nick Nolte
9 Lindsay Lohan
10 Al Gore

Schema	*Customer*
Tenant 1 *(Hertz)*	*1* Lance Armstrong
	2 John Cleese
	3 Tipper Gore
Tenant 2 *(Hertz)*	*4* Shaquille O'Neal
	7 Ben Stein
	8 William Shatner
Tenant 3 *(Rent-a-wreck)*	*2* Dennis Rodman
	7 Nick Nolte
	9 Lindsay Lohan
	10 Al Gore

Benefits of “the best thing since sliced bread"

Simplifies development

- Minimal application changes
- No tenant-based customizations for queries or other data access

Eases deployment

- Tenant access to data is transparent, based on identity
- Tenants can be quickly and efficiently added, removed, and managed

Decreases maintenance overhead

- Fewer databases to manage, better resource utilization
- Tenant-based utilities and tools make maintenance tasks easier

Maintains security of tenant data

- Physical separation within database
- Tenant authentication required for data access

All Questions answered

PROGRESS