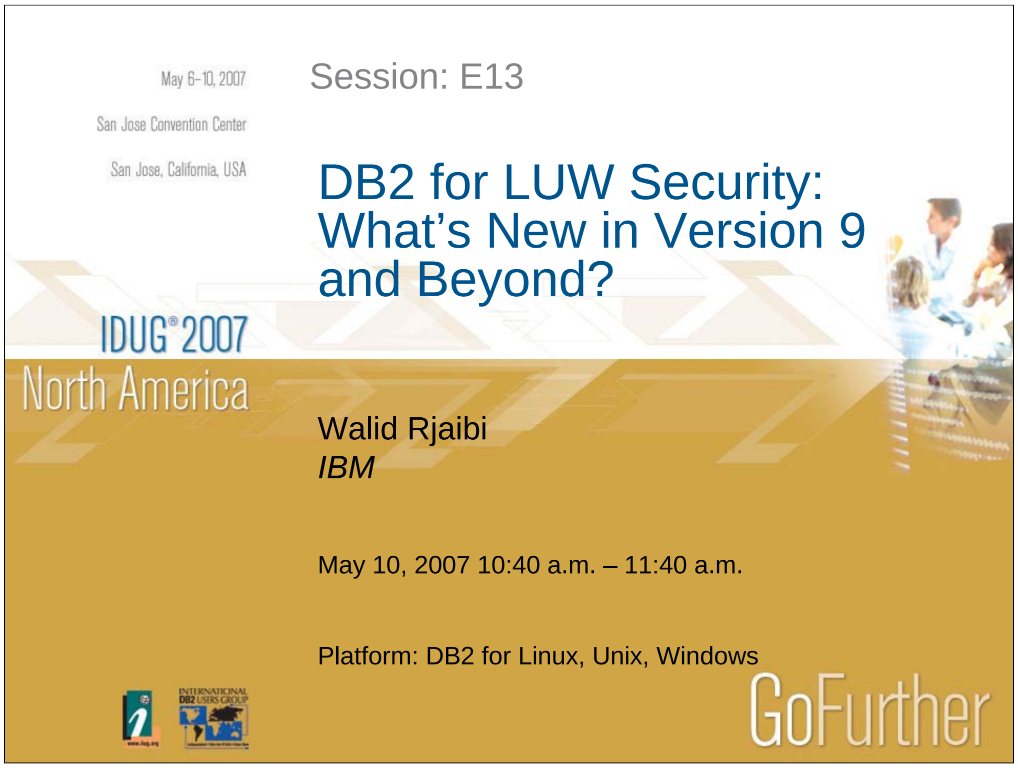


May 6–10, 2007

San Jose Convention Center

San Jose, California, USA

IDUG® 2007 North America

Session: E13

DB2 for LUW Security: What's New in Version 9 and Beyond?

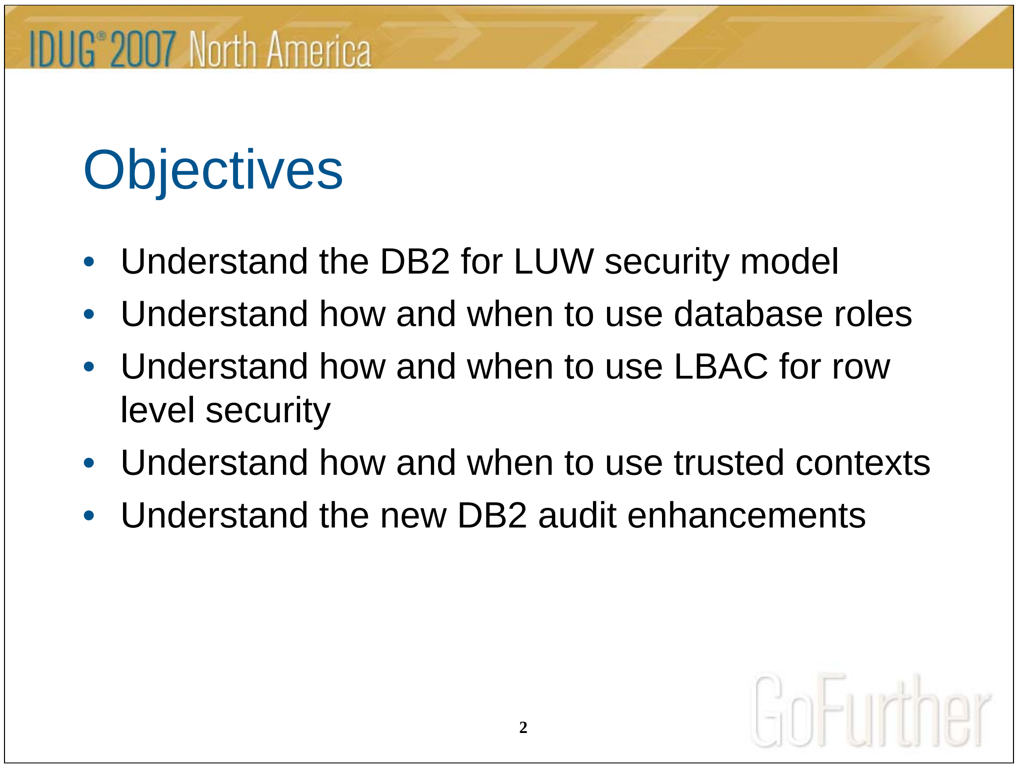
Walid Rjaibi
IBM

May 10, 2007 10:40 a.m. – 11:40 a.m.

Platform: DB2 for Linux, Unix, Windows

GoFurther

Securing information assets and restricting data access to only those with the need to know is becoming a significant challenge for many organizations today. DB2 for LUW Security has been enriched with several new capabilities since version 8. This presentation will give an overview of the new security capabilities introduced in version 9 as well as a look ahead into the upcoming security capabilities in DB2 VIPER II.



Objectives

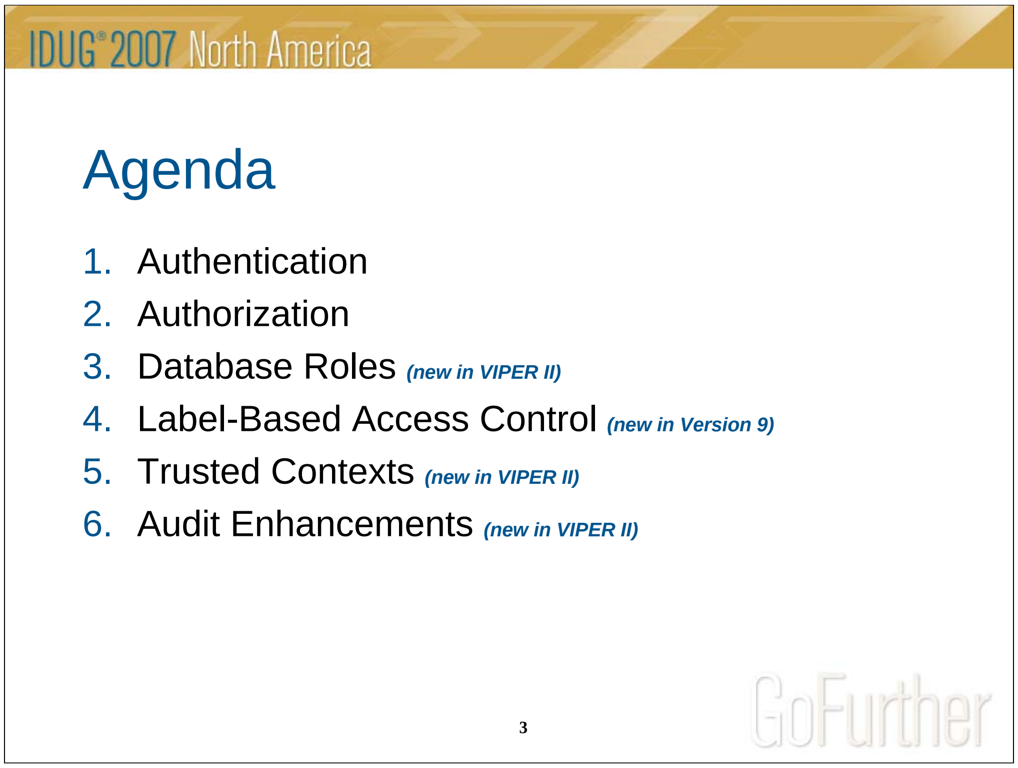
- Understand the DB2 for LUW security model
- Understand how and when to use database roles
- Understand how and when to use LBAC for row level security
- Understand how and when to use trusted contexts
- Understand the new DB2 audit enhancements

2

GoFurther

The presentation will cover the following topics:

1. Authentication
2. Authorization
3. Database Roles *(new in VIPER II)*
4. Label-Based Access Control *(new in Version 9)*
5. Trusted Contexts *(new in VIPER II)*
6. Audit Enhancements *(new in VIPER II)*



IDUG® 2007 North America

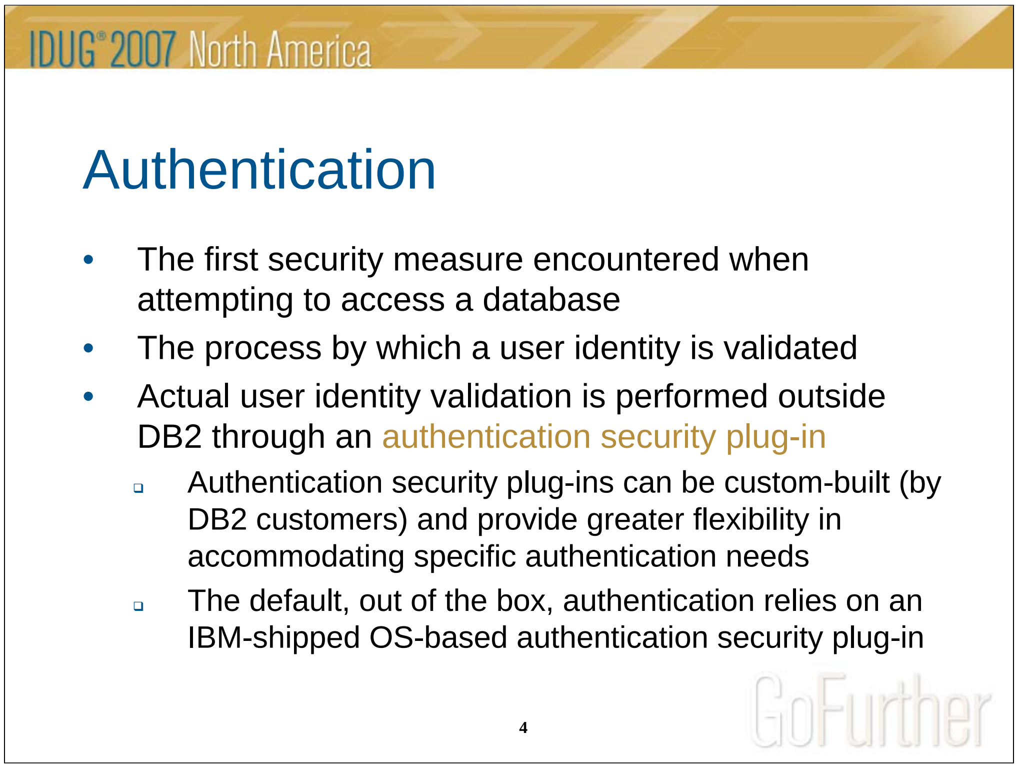
Agenda

1. Authentication
2. Authorization
3. Database Roles ***(new in VIPER II)***
4. Label-Based Access Control ***(new in Version 9)***
5. Trusted Contexts ***(new in VIPER II)***
6. Audit Enhancements ***(new in VIPER II)***

3

GoFurther

This is the presentation outline.



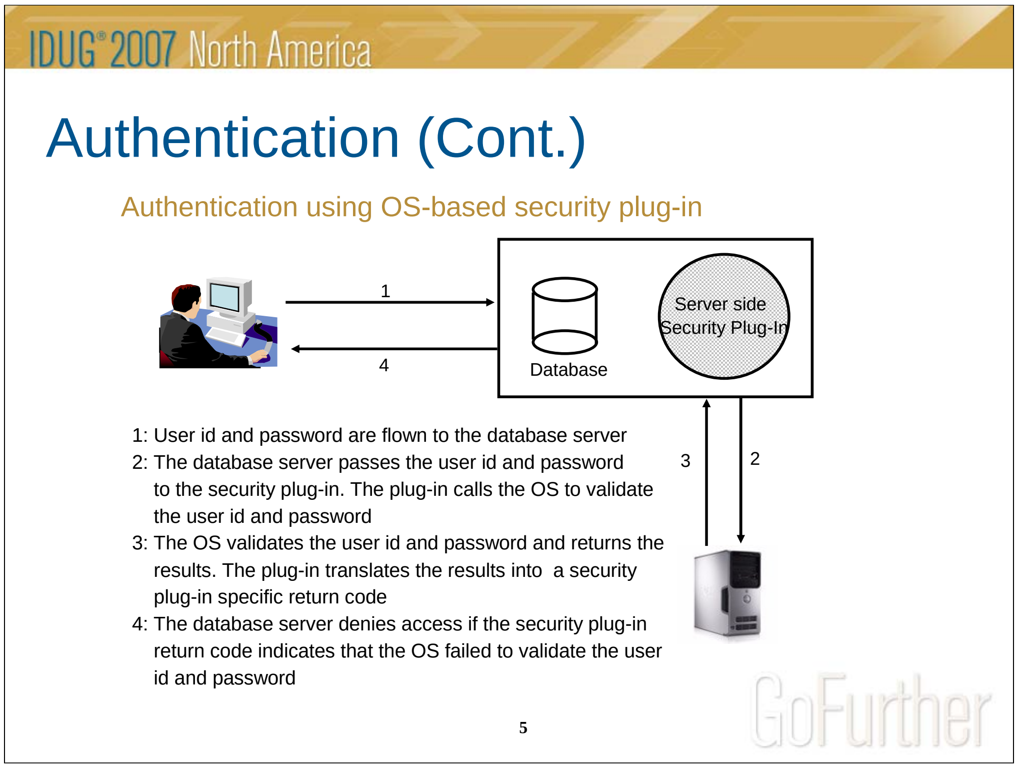
Authentication

- The first security measure encountered when attempting to access a database
- The process by which a user identity is validated
- Actual user identity validation is performed outside DB2 through an authentication security plug-in
    - Authentication security plug-ins can be custom-built (by DB2 customers) and provide greater flexibility in accommodating specific authentication needs
    - The default, out of the box, authentication relies on an IBM-shipped OS-based authentication security plug-in

Authentication is the first DB2 security feature encountered when a user attempts to connect to a database or to attach to a DB2 instance.

This is the process whereby the user identity is validated before access is granted or rejected. The actual user validation is performed outside DB2 through an authentication security plug-in.

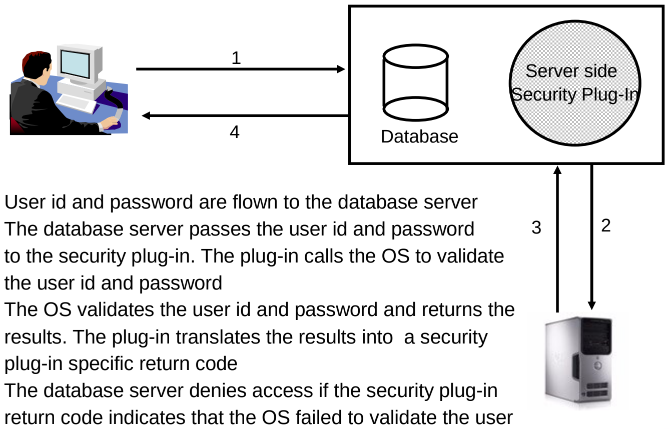
Authentication security plug-ins can be custom-built by DB2 customers to accommodate specific authentication needs. The default, out of the box, authentication relies on an IBM-shipped OS-based authentication security plug-in.



IDUG® 2007 North America

Authentication (Cont.)

Authentication using OS-based security plug-in



1: User id and password are flown to the database server
2: The database server passes the user id and password to the security plug-in. The plug-in calls the OS to validate the user id and password
3: The OS validates the user id and password and returns the results. The plug-in translates the results into a security plug-in specific return code
4: The database server denies access if the security plug-in return code indicates that the OS failed to validate the user id and password

5

GoFurther

The diagram above illustrates the authentication process when an OS-based authentication security plug-in is used.

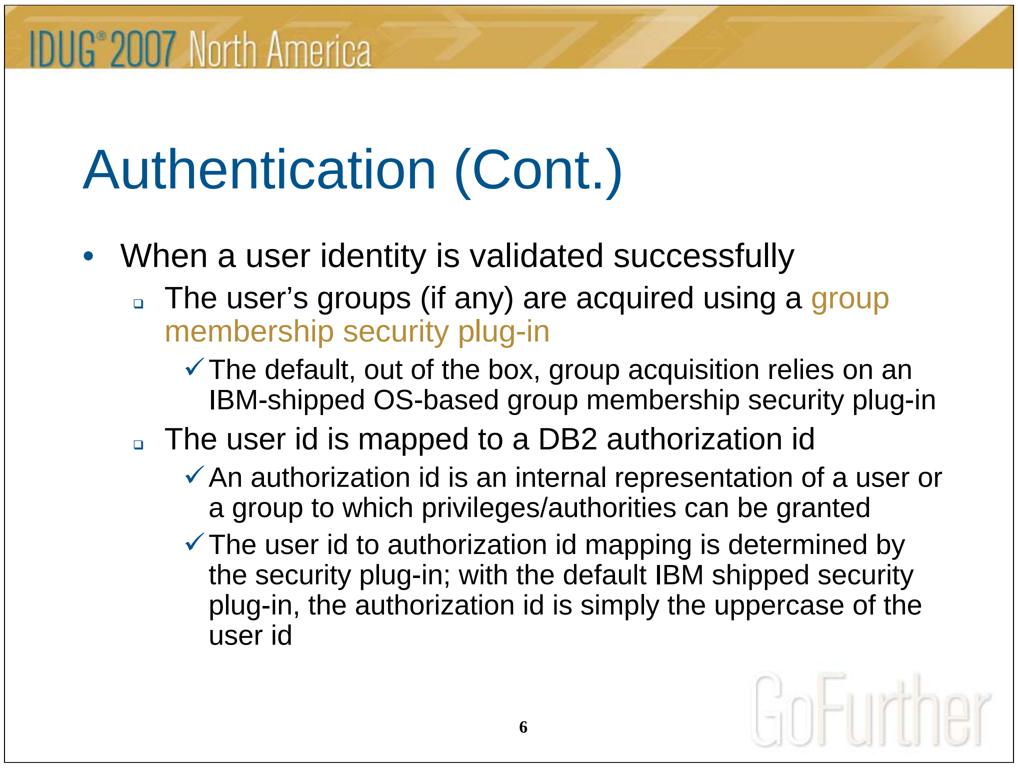
There are 4 steps to the authentication process:

1: The user id and password are flown from the client to the database server

2: The database server passes the user id and password to the security plug-in. The plug-in calls the OS to validate the user id and password

3: The OS validates the user id and password and returns the results. The security plug-in translates these results into a security plug-in specific return code

4: The database server denies access if the security plug-in return code indicates that the OS failed to validate the user id and password. If the OS validates the user id and password successfully, then access to the database is granted if the user holds CONNECT privilege to the database. In other words, passing the authentication step alone is not sufficient to gain access to the database.


IDUG® 2007 North America

Authentication (Cont.)

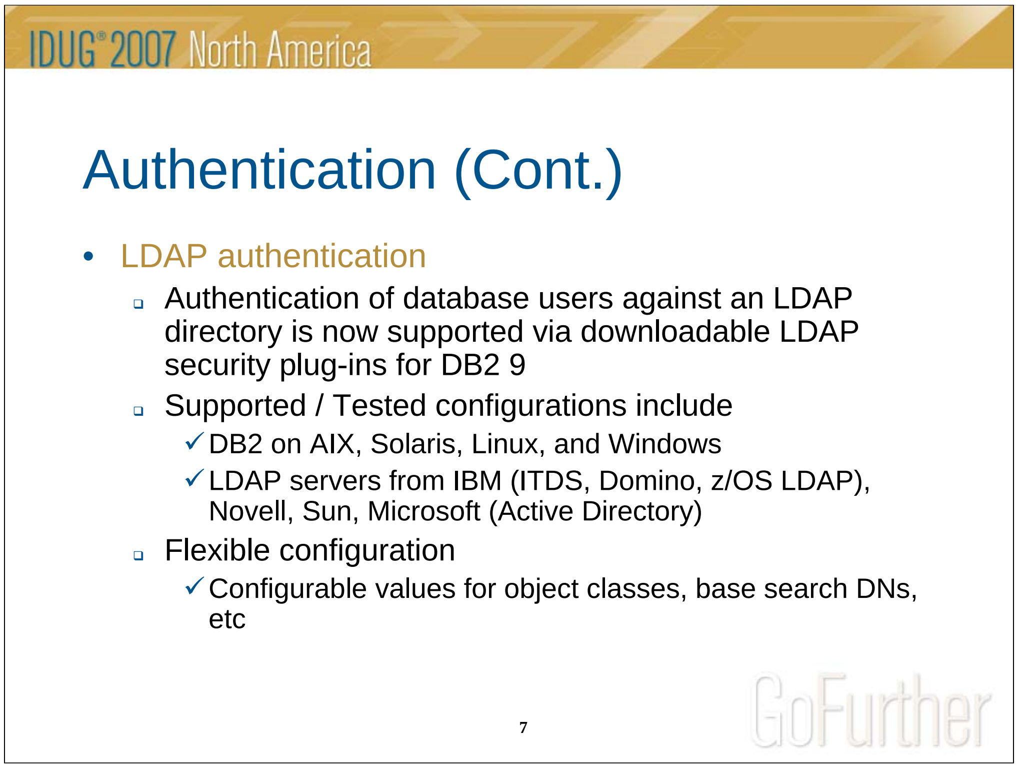
- When a user identity is validated successfully
  - The user's groups (if any) are acquired using a group membership security plug-in
    - ✓ The default, out of the box, group acquisition relies on an IBM-shipped OS-based group membership security plug-in
  - The user id is mapped to a DB2 authorization id
    - ✓ An authorization id is an internal representation of a user or a group to which privileges/authorities can be granted
    - ✓ The user id to authorization id mapping is determined by the security plug-in; with the default IBM shipped security plug-in, the authorization id is simply the uppercase of the user id

6

GoFurther

When DB2 authenticates a user successfully, the user's groups (if any) are acquired using a group membership security plug-in. Groups are not defined or managed by DB2. They are defined and managed by an external entity such as an operating system or an LDAP directory. The default, out of the box, group acquisition relies on an IBM-shipped OS-based group membership security plug-in.

Lastly, the authenticated user ID is mapped to a DB2 authorization ID. This mapping is determined by the authentication security plug-in. When using the default IBM shipped authentication security plug-in, the authorization id is simply the uppercase of the user id. This derived authorization ID is also referred to as the system authorization ID or the primary authorization ID, and is the one checked by DB2 when performing authorization checks.


IDUG® 2007 North America
GoFurther

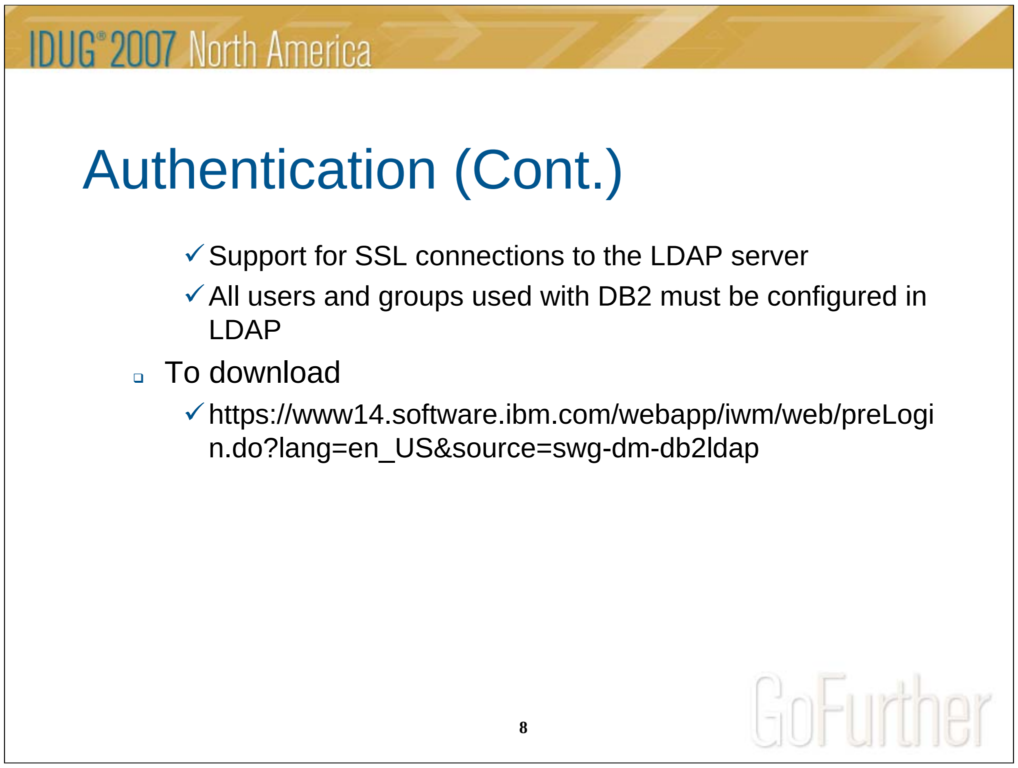
Authentication (Cont.)

- LDAP authentication
  - Authentication of database users against an LDAP directory is now supported via downloadable LDAP security plug-ins for DB2 9
  - Supported / Tested configurations include
    - ✓ DB2 on AIX, Solaris, Linux, and Windows
    - ✓ LDAP servers from IBM (ITDS, Domino, z/OS LDAP), Novell, Sun, Microsoft (Active Directory)
  - Flexible configuration
    - ✓ Configurable values for object classes, base search DNs, etc

7

Authentication via Lightweight Directory Access Protocol (LDPA) is now supported via downloadable LDAP security plug-ins for DB2 9. The following configurations are tested and supported:

1. DB2 on AIX, Solaris, Linux, and Windows
2. LDAP servers from IBM (ITDS, Domino, z/OS LDAP), Novell, Sun, Microsoft (Active Directory)


IDUG® 2007 North America

Authentication (Cont.)

- ✓ Support for SSL connections to the LDAP server
- ✓ All users and groups used with DB2 must be configured in LDAP

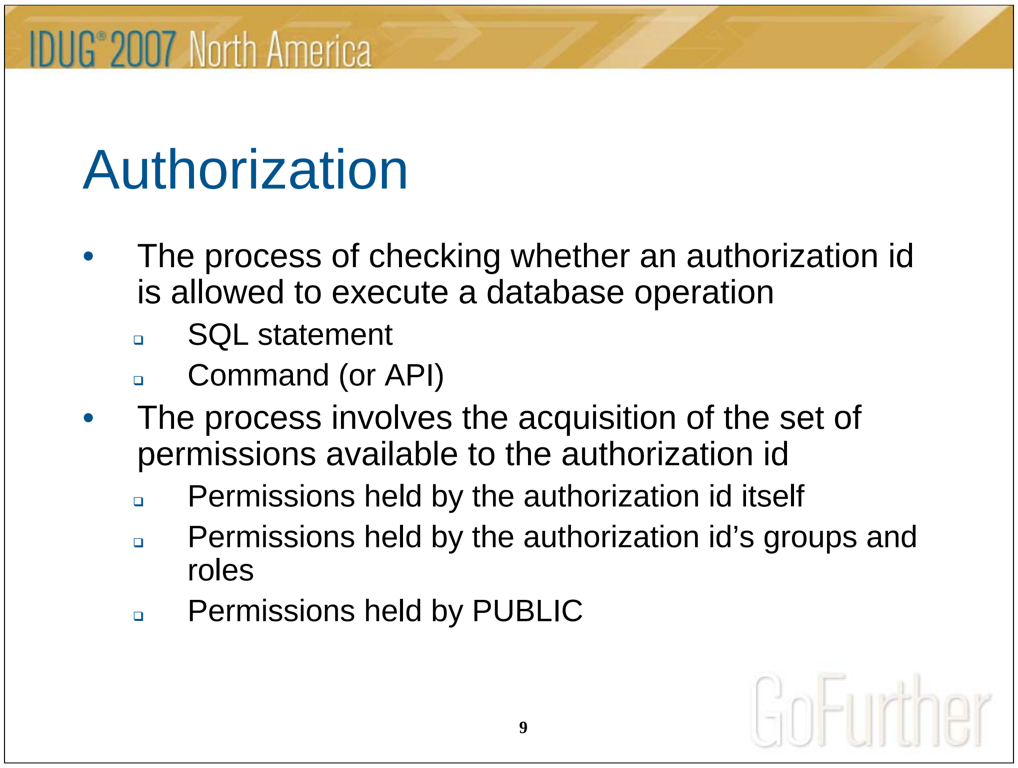
- ❑ To download
  - ✓ https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2ldap

8

GoFurther

To download the LDAP security plug-ins for DB2 version 9, please use the following URL:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2ldap


IDUG® 2007 North America

Authorization

- The process of checking whether an authorization id is allowed to execute a database operation
  - SQL statement
  - Command (or API)
- The process involves the acquisition of the set of permissions available to the authorization id
  - Permissions held by the authorization id itself
  - Permissions held by the authorization id's groups and roles
  - Permissions held by PUBLIC

9

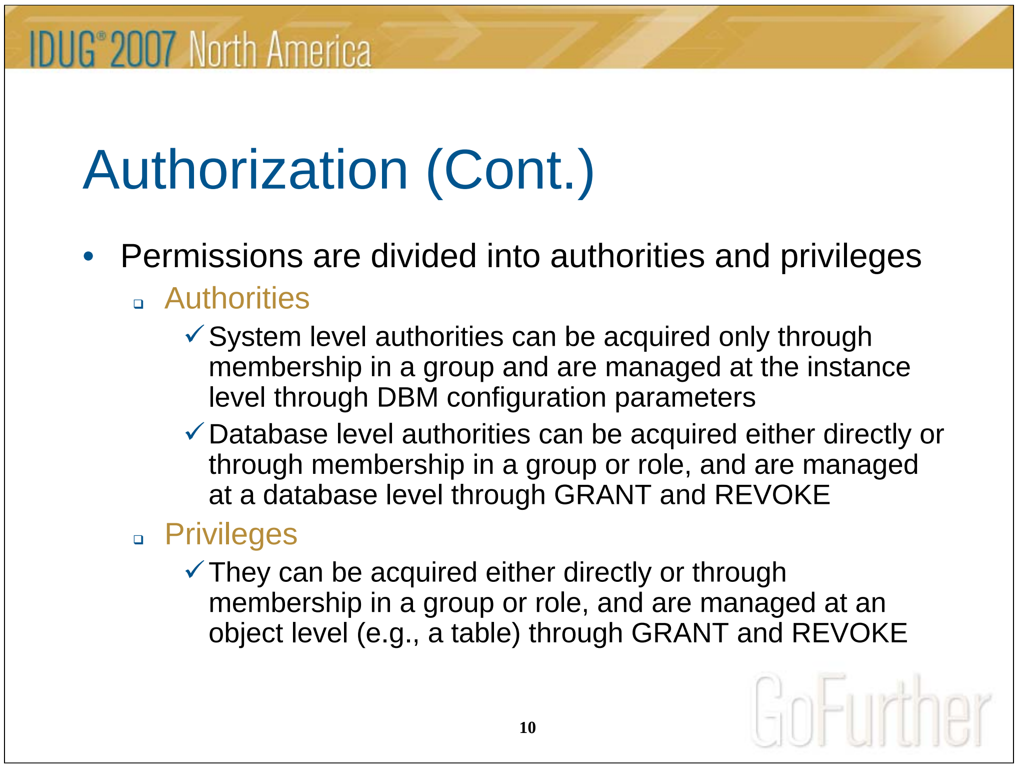
GoFurther

Authorization is the process of checking whether an authorization ID (i.e., an already authenticated user) is allowed to execute a database operation such as an SQL statement, a command, or an API.

To determine whether or not to allow a user to execute a database operation, the authorization process checks if the user has been granted a permission to execute that operation.

At any point in time, the permissions available to an authorization ID are the union of:

1. The permissions granted to the authorization ID itself
2. The permissions granted to the authorization id's groups and roles
3. The permissions granted to the special group PUBLIC (which includes all users)



IDUG® 2007 North America

Authorization (Cont.)

- Permissions are divided into authorities and privileges
  - Authorities
    - ✓ System level authorities can be acquired only through membership in a group and are managed at the instance level through DBM configuration parameters
    - ✓ Database level authorities can be acquired either directly or through membership in a group or role, and are managed at a database level through GRANT and REVOKE
  - Privileges
    - ✓ They can be acquired either directly or through membership in a group or role, and are managed at an object level (e.g., a table) through GRANT and REVOKE

10

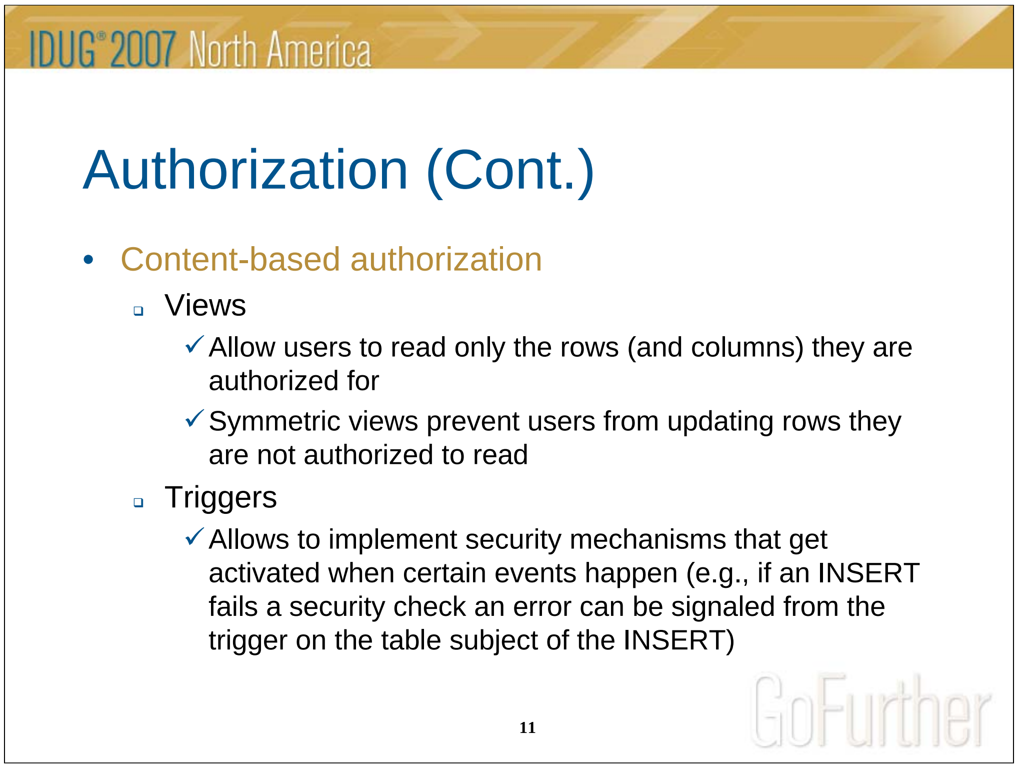
GoFurther

Permissions are divided into two categories: Authorities and Privileges.

Authorities are also divided into 2 categories: System level authorities and database level authorities. System level authorities can be acquired only through membership in a group and are managed at the instance level through DBM configuration parameters. Database level authorities can be acquired either directly or through membership in a group or role, and are managed at a database level through GRANT and REVOKE.

Privileges can be acquired either directly or through membership in a group or role, and are managed at an object level (e.g., a table) through GRANT and REVOKE. For example, the DBA can choose to grant user Joe the privilege to select from the table EMPLOYEE. He would do so by executing the following SQL statement:

```
GRANT SELECT ON EMPLOYEE TO USER Joe
```



IDUG® 2007 North America

Authorization (Cont.)

- Content-based authorization
  - Views
    - ✓ Allow users to read only the rows (and columns) they are authorized for
    - ✓ Symmetric views prevent users from updating rows they are not authorized to read
  - Triggers
    - ✓ Allows to implement security mechanisms that get activated when certain events happen (e.g., if an INSERT fails a security check an error can be signaled from the trigger on the table subject of the INSERT)

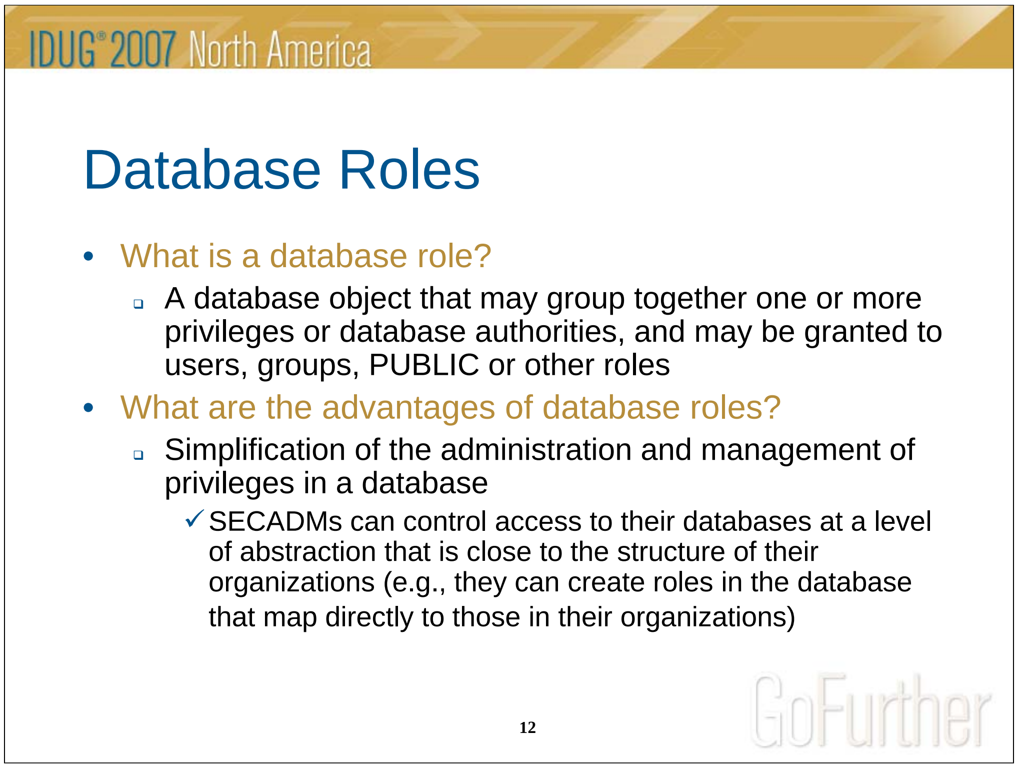
11

GoFurther

Content-based authorization can be implemented using views and triggers, for example, to allow users to read only the rows (and columns) they are authorized for.

For even greater flexibility, symmetric views can be used to prevent users from updating rows they are not authorized to read.

Triggers allow to implement security mechanisms that get activated when certain events happen (e.g., if an INSERT fails a security check an error can be signaled from the trigger on the table subject of the INSERT).


IDUG® 2007 North America

Database Roles

- What is a database role?
  - A database object that may group together one or more privileges or database authorities, and may be granted to users, groups, PUBLIC or other roles
- What are the advantages of database roles?
  - Simplification of the administration and management of privileges in a database
    - ✓ SECADMs can control access to their databases at a level of abstraction that is close to the structure of their organizations (e.g., they can create roles in the database that map directly to those in their organizations)

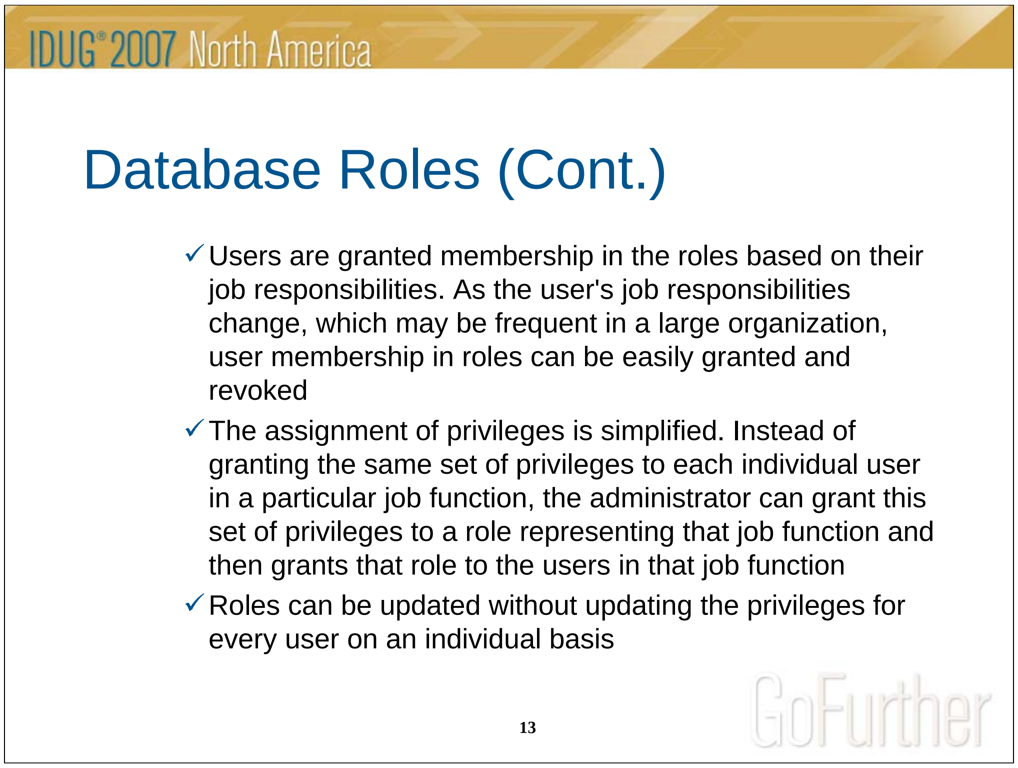
12

GoFurther

Database roles can be thought of as groups that are defined and managed by DB2. A database role is essentially a database object that may group together one or more privileges or database authorities, and may be granted to users, groups, PUBLIC or other roles.

The key advantage of database roles is that they simplify the administration and management of privileges in a database. For instance:

1. Security administrators can control access to their databases at a level of abstraction that is close to the structure of their organizations (e.g., they can create roles in the database that map directly to those in their organizations)
2. Users are granted membership in the roles based on their job responsibilities. As the user's job responsibilities change, which may be frequent in a large organization, user membership in roles can be easily granted and revoked
3. The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role representing that job function and then grants that role to the users in that job function
4. Roles can be updated without updating the privileges for every user on an individual basis


IDUG® 2007 North America

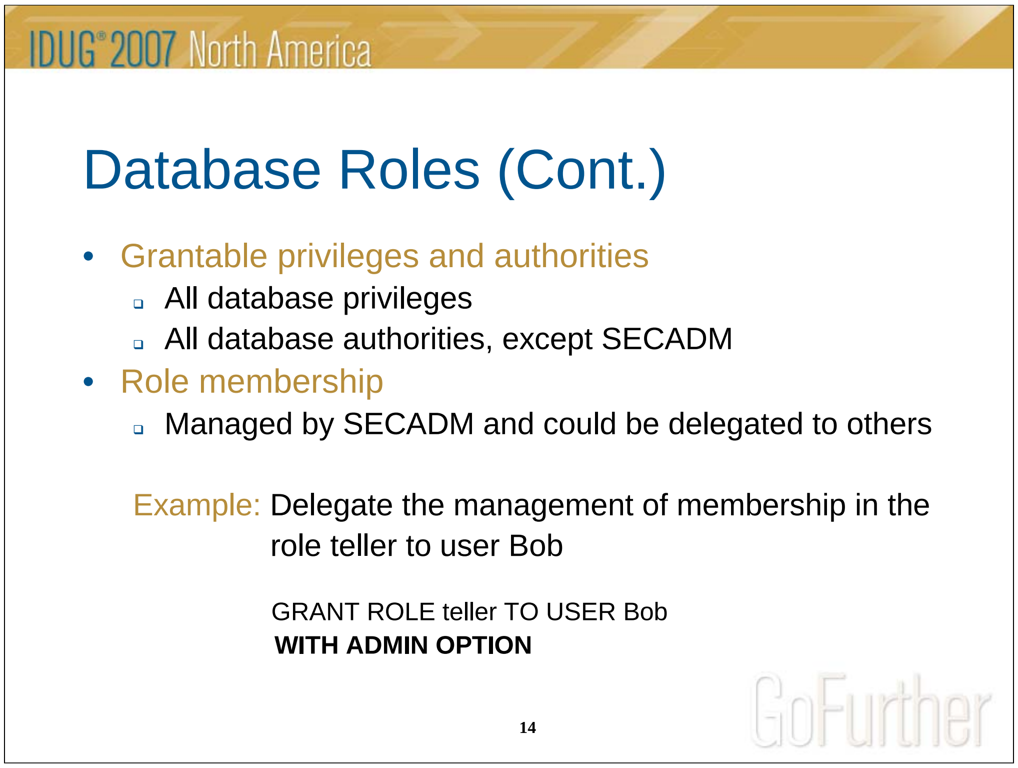
Database Roles (Cont.)

- ✓ Users are granted membership in the roles based on their job responsibilities. As the user's job responsibilities change, which may be frequent in a large organization, user membership in roles can be easily granted and revoked
- ✓ The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role representing that job function and then grants that role to the users in that job function
- ✓ Roles can be updated without updating the privileges for every user on an individual basis

13

GoFurther

In essence, by granting access permissions to roles only and by making users members in roles, the administration and management of privileges in the database is greatly simplified.


IDUG® 2007 North America

Database Roles (Cont.)

- Grantable privileges and authorities
  - All database privileges
  - All database authorities, except SECADM
- Role membership
  - Managed by SECADM and could be delegated to others

Example: Delegate the management of membership in the role teller to user Bob

```
GRANT ROLE teller TO USER Bob
WITH ADMIN OPTION
```

14

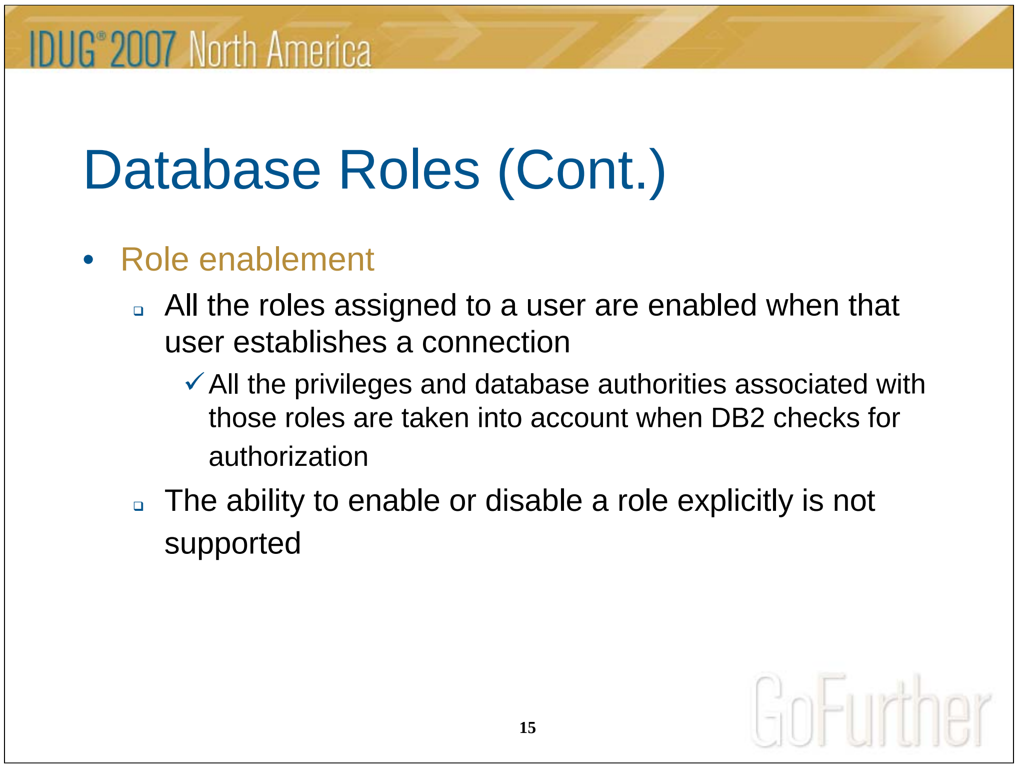
GoFurther

Any privilege or authority (except SECADM) grantable within the database can be granted to a role.

The authority to manage membership in a role is vested in the security administrator (SECADM). The SECADM uses the GRANT and REVOKE SQL statements to add a user to a role or to remove a user from a role.

If they so desire, the security administrator can also delegate the management of membership in a role to some other user. This is possible by specifying the WITH ADMIN option on the GRANT ROLE SQL statement.

Users who hold the WITH ADMIN option on a role can ONLY manage membership in the role, i.e., they cannot drop the role. Only the SECADM can do so.



IDUG® 2007 North America

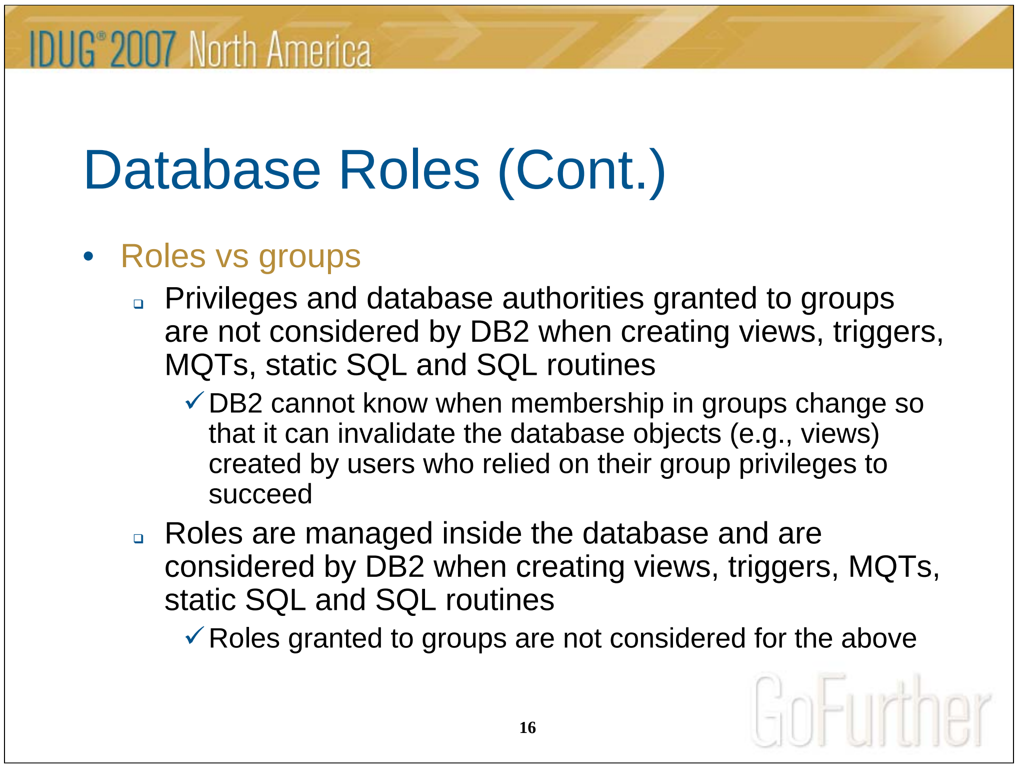
Database Roles (Cont.)

- Role enablement
  - All the roles assigned to a user are enabled when that user establishes a connection
    - ✓ All the privileges and database authorities associated with those roles are taken into account when DB2 checks for authorization
  - The ability to enable or disable a role explicitly is not supported

15

GoFurther

All the roles assigned to a user are available to them from the moment they establish a connection. The SET ROLE SQL statement is supported but it is really a no-op. It cannot be used to enable or disable a role.


IDUG® 2007 North America

Database Roles (Cont.)

- Roles vs groups
  - Privileges and database authorities granted to groups are not considered by DB2 when creating views, triggers, MQTs, static SQL and SQL routines
    - ✓ DB2 cannot know when membership in groups change so that it can invalidate the database objects (e.g., views) created by users who relied on their group privileges to succeed
  - Roles are managed inside the database and are considered by DB2 when creating views, triggers, MQTs, static SQL and SQL routines
    - ✓ Roles granted to groups are not considered for the above

16

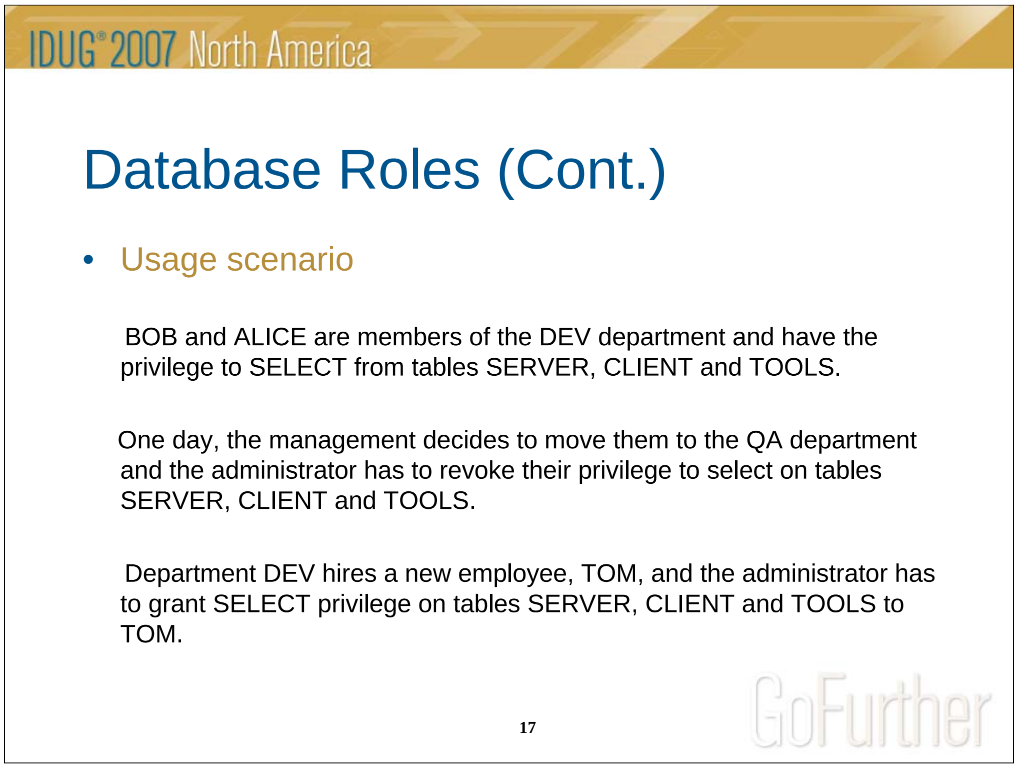
GoFurther

Another key advantage of using roles vs groups in DB2 is that roles are managed by the database. That is, DB2 knows when membership in a role changes.

This is not the case with groups because they are managed outside the database (e.g., by the operating system). Because of this, DB2 imposes some restrictions on when privileges and authorities assigned to groups are taken into account when performing an authorization check. These restrictions include not taking groups into account when creating the following objects:

1. Views
2. Triggers
3. Materialized Query Tables (MQTs)
4. Packages with static SQL
5. SQL routines

Because roles are managed by DB2, the restriction above does not apply to roles. That is, privileges granted to roles are taken into account when creating the above objects. Roles granted to groups are not considered for those objects, however.



IDUG® 2007 North America

Database Roles (Cont.)

- Usage scenario

BOB and ALICE are members of the DEV department and have the privilege to SELECT from tables SERVER, CLIENT and TOOLS.

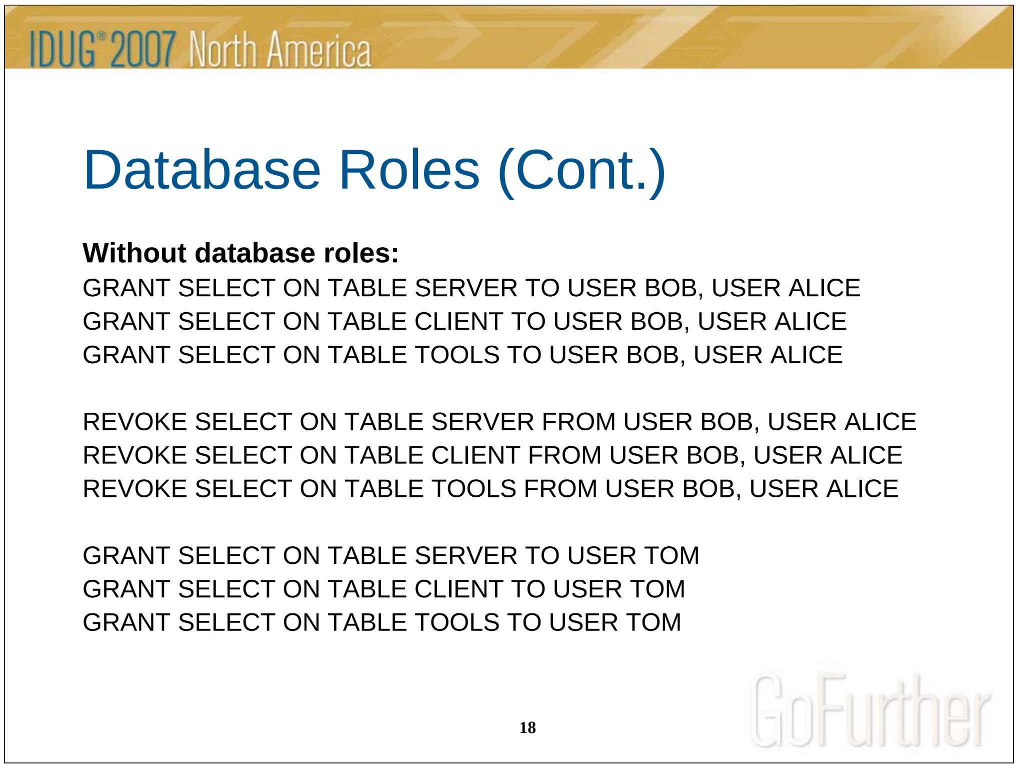
One day, the management decides to move them to the QA department and the administrator has to revoke their privilege to select on tables SERVER, CLIENT and TOOLS.

Department DEV hires a new employee, TOM, and the administrator has to grant SELECT privilege on tables SERVER, CLIENT and TOOLS to TOM.

17

GoFurther

The above is a hypothetical usage scenario to illustrate how the management of privileges in a database can be simplified if access permissions are associated with roles and users are simply made members in those roles.



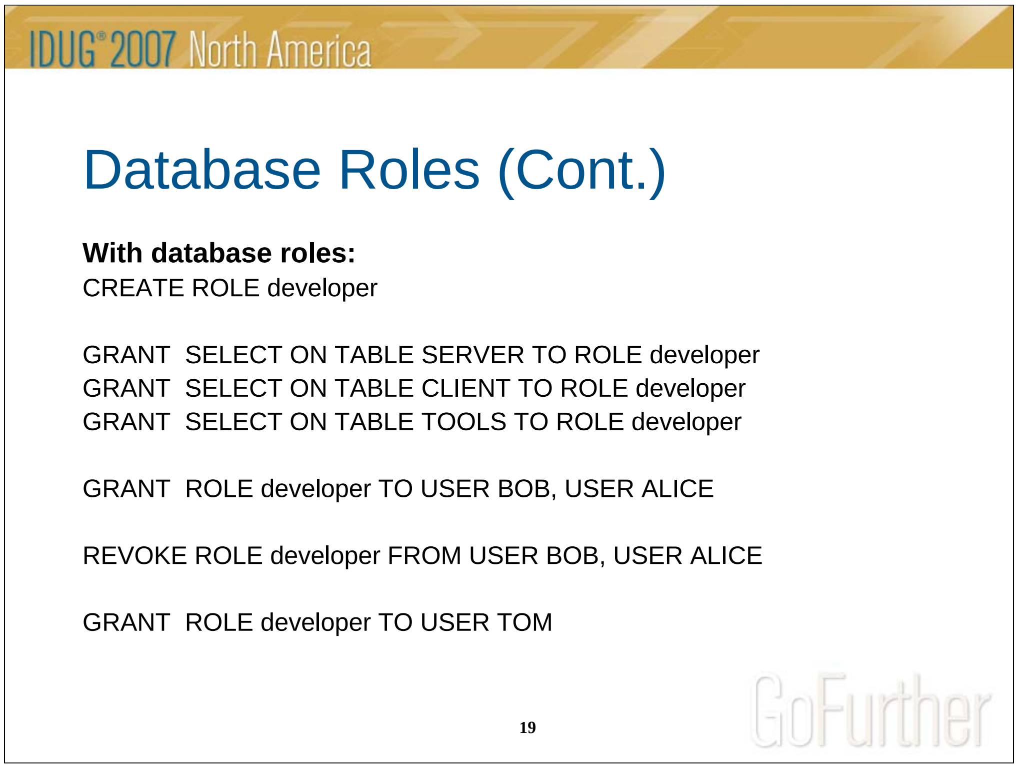
Database Roles (Cont.)

Without database roles:

```
GRANT SELECT ON TABLE SERVER TO USER BOB, USER ALICE
GRANT SELECT ON TABLE CLIENT TO USER BOB, USER ALICE
GRANT SELECT ON TABLE TOOLS TO USER BOB, USER ALICE

REVOKE SELECT ON TABLE SERVER FROM USER BOB, USER ALICE
REVOKE SELECT ON TABLE CLIENT FROM USER BOB, USER ALICE
REVOKE SELECT ON TABLE TOOLS FROM USER BOB, USER ALICE

GRANT SELECT ON TABLE SERVER TO USER TOM
GRANT SELECT ON TABLE CLIENT TO USER TOM
GRANT SELECT ON TABLE TOOLS TO USER TOM
```

GoFurther

Without database roles, access permissions are directly associated with users. When there is a need to revoke access permissions from a user, the access permissions has to be revoked from the user explicitly. Note that there are 9 SQL statements in total here.


IDUG® 2007 North America

Database Roles (Cont.)

With database roles:

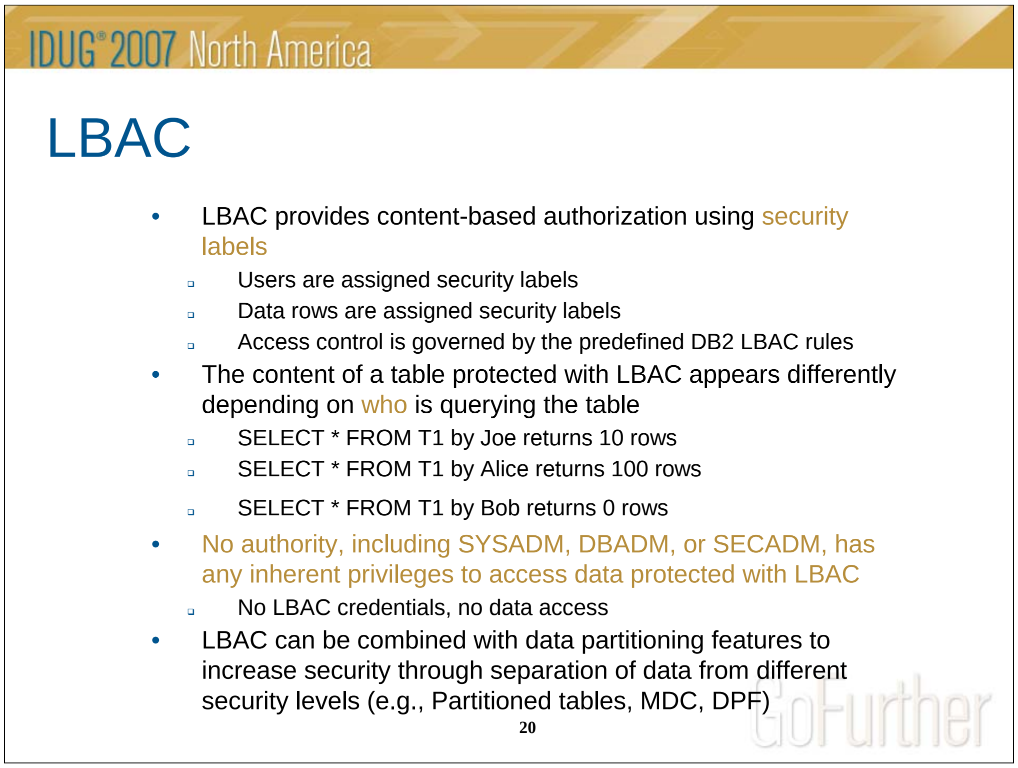
```
CREATE ROLE developer

GRANT  SELECT ON TABLE SERVER TO ROLE developer
GRANT  SELECT ON TABLE CLIENT TO ROLE developer
GRANT  SELECT ON TABLE TOOLS TO ROLE developer

GRANT  ROLE developer TO USER BOB, USER ALICE

REVOKE ROLE developer FROM USER BOB, USER ALICE

GRANT  ROLE developer TO USER TOM
```

19

GoFurther

With database roles, access permissions are associated with roles, not with users. Users are simply made members in a role. When there is a need to restrict access for a particular user, all that is required is to revoke their membership in the role.

Note that there are 7 SQL statements in total here, compared to 9 in the previous example without database roles. The savings could be much bigger for real scenarios dealing with possibly dozens of users.


IDUG® 2007 North America

LBAC

- LBAC provides content-based authorization using security labels
  - Users are assigned security labels
  - Data rows are assigned security labels
  - Access control is governed by the predefined DB2 LBAC rules
- The content of a table protected with LBAC appears differently depending on who is querying the table
  - SELECT * FROM T1 by Joe returns 10 rows
  - SELECT * FROM T1 by Alice returns 100 rows
  - SELECT * FROM T1 by Bob returns 0 rows
- No authority, including SYSADM, DBADM, or SECADM, has any inherent privileges to access data protected with LBAC
  - No LBAC credentials, no data access
- LBAC can be combined with data partitioning features to increase security through separation of data from different security levels (e.g., Partitioned tables, MDC, DPF)

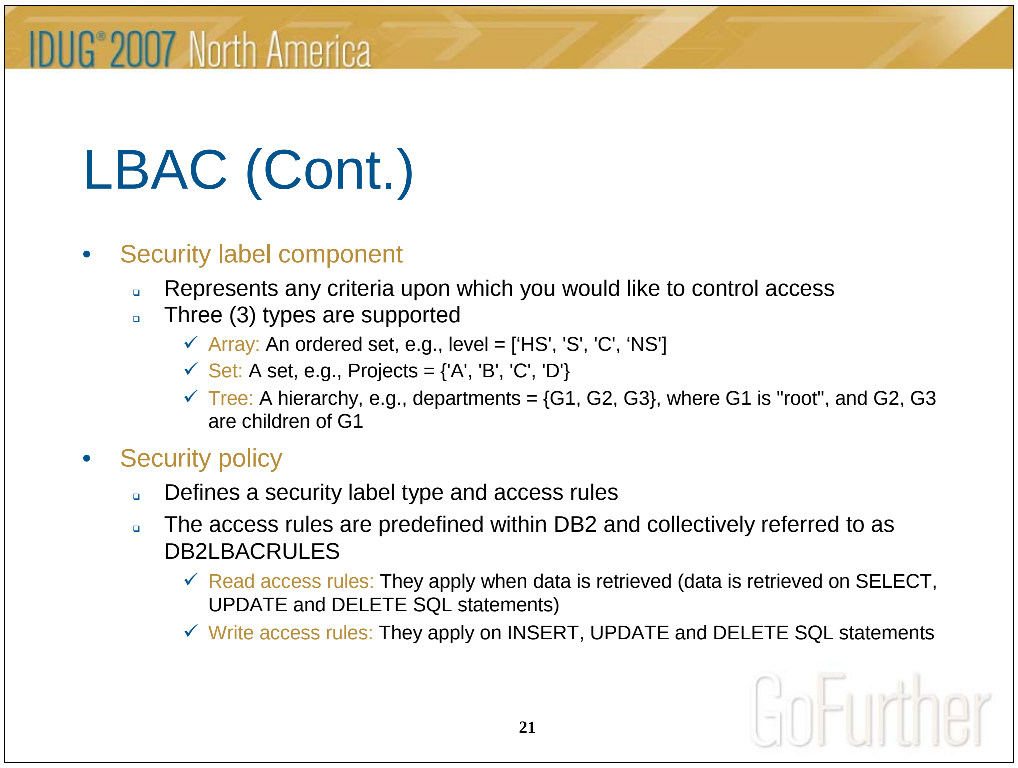
20

GoFurther

Label-Based access control (LBAC) allows you to control access to your data at the row level based on security labels. With LBAC, you assign security labels to your users and to your data rows. DB2 will apply the DB2 LBAC rules when a user accesses a table protected with LBAC. The rules could allow the user to see all the rows, some of the rows, or none of the rows depending on their security label and the security labels protecting the data rows in the table.

One key advantage of using LBAC to protect sensitive data is that no authority, including SYSADM, DBADM, or SECADM, has any inherent privileges to access data protected with LBAC.

Another advantage is that LBAC can be easily combined with data partitioning features available in DB2 to further increase security. For example, by combining LBAC and Data Partitioning Facility (DPF), you could choose to separate data from different security levels such that your highly sensitive data is on your most secure partition, your sensitive data is on your second most secure partition and so on.


IDUG® 2007 North America

LBAC (Cont.)

- Security label component
  - Represents any criteria upon which you would like to control access
  - Three (3) types are supported
    - ✓ Array: An ordered set, e.g., level = ['HS', 'S', 'C', 'NS']
    - ✓ Set: A set, e.g., Projects = {'A', 'B', 'C', 'D'}
    - ✓ Tree: A hierarchy, e.g., departments = {G1, G2, G3}, where G1 is "root", and G2, G3 are children of G1
- Security policy
  - Defines a security label type and access rules
  - The access rules are predefined within DB2 and collectively referred to as DB2LBACRULES
    - ✓ Read access rules: They apply when data is retrieved (data is retrieved on SELECT, UPDATE and DELETE SQL statements)
    - ✓ Write access rules: They apply on INSERT, UPDATE and DELETE SQL statements

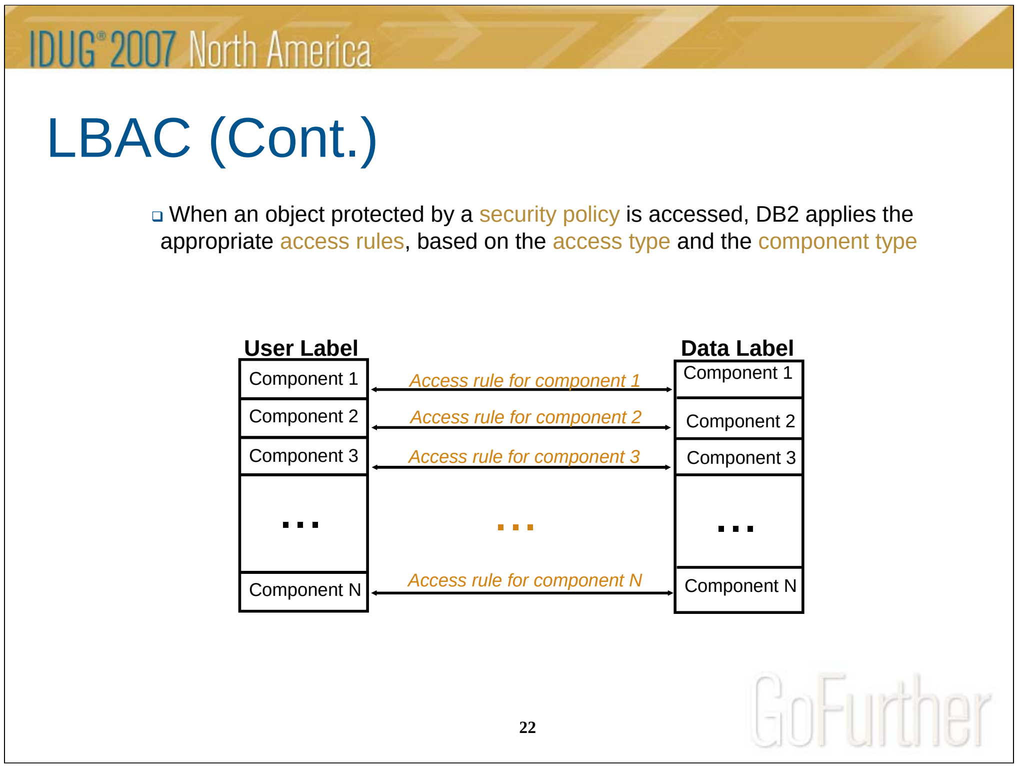
21

GoFurther

The key DB2 LBAC concepts are security label components, security policies and security labels.

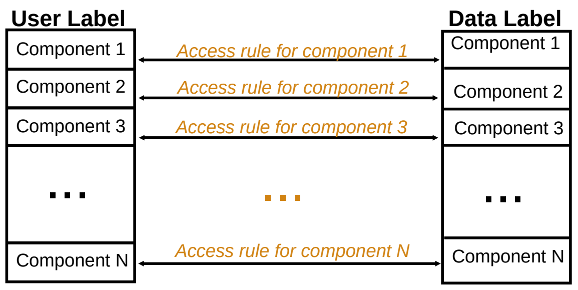
A security label component represents any criteria upon which you would like to base access control. For example, you may want to base access control based on the color associated with the row. In this case you would define a security label component of type set that contains a set of colors such as {red, green, blue}.

Security label components are building blocks for security policies. A security policy defines the structure of a security label. That is, it defines the number and type of components that would make up such security label. For example, one could define a security policy that consists of a single component such as the color component above. So, in this case, security labels will consist of colors. You can assign colors to your rows and colors to your users. A user with the blue color can see all blue rows, but not the red or the green rows. A User with a security label that contains both blue and red will be able to see all the blue rows and all the red rows, but not the green rows.



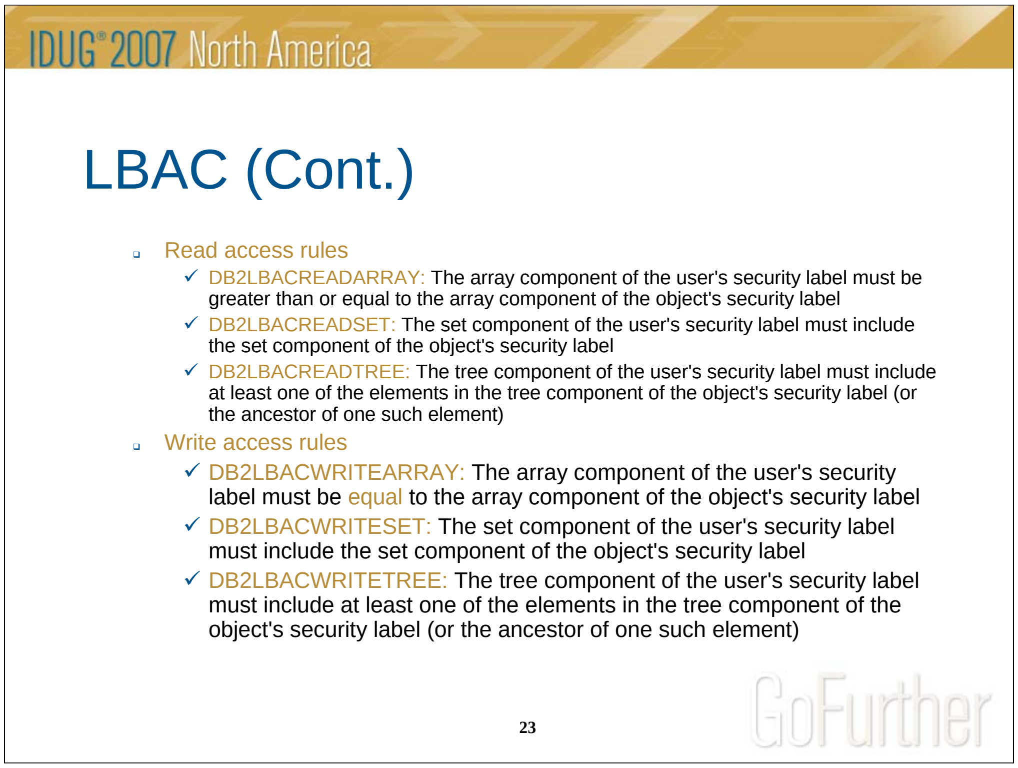
LBAC (Cont.)

- When an object protected by a security policy is accessed, DB2 applies the appropriate access rules, based on the access type and the component type



One key advantage of the DB2 LBAC implementation compared to classical Multilevel Security (MLS) implementations is its inherent flexibility. That is, the flexibility to define a security policy with the number and type of components that best suit your needs. Classical MLS implementations come with a rigid hardwired security policy with 2 components: a level (equivalent to arrays in DB2 LBAC) and compartments (equivalent to sets in DB2 LBAC).

When comparing two security labels for access control purposes, DB2 looks at the security policy definition and selects the appropriate rule to apply given the type of the current security label component. When the number of components in the security policy is exhausted and all associated rules are satisfied, access is granted. Otherwise, access is rejected.


IDUG® 2007 North America

LBAC (Cont.)

- Read access rules
  - ✓ DB2LBACREADARRAY: The array component of the user's security label must be greater than or equal to the array component of the object's security label
  - ✓ DB2LBACREADSET: The set component of the user's security label must include the set component of the object's security label
  - ✓ DB2LBACREADTREE: The tree component of the user's security label must include at least one of the elements in the tree component of the object's security label (or the ancestor of one such element)
- Write access rules
  - ✓ DB2LBACWRITEARRAY: The array component of the user's security label must be equal to the array component of the object's security label
  - ✓ DB2LBACWRITESET: The set component of the user's security label must include the set component of the object's security label
  - ✓ DB2LBACWRITETREE: The tree component of the user's security label must include at least one of the elements in the tree component of the object's security label (or the ancestor of one such element)

23

GoFurther

The predefined DB2 LBAC rules are divided into two categories: The read access rules and the write access rules.

The read access rules are applied whenever data is fetched from a table protected with LBAC. For example, data is fetched during a SELECT SQL statement. Data is also fetched during the fetch phase of an UPDATE or a DELETE SQL statement.

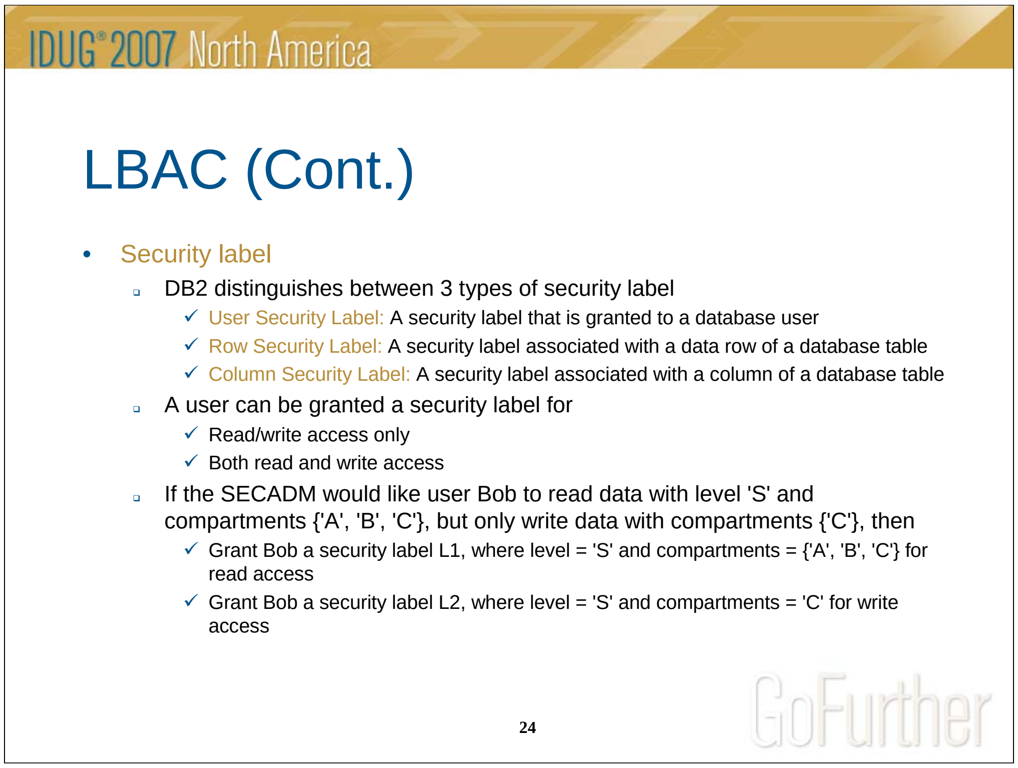
The write access rules apply when data in a protected table is modified. For example, data is modified during an INSERT SQL statement.

The read access rules are divided into three categories:

1. DB2LBACREADARRAY which applies when comparing two security label components of type array for a read operation
2. DB2LBACREADSET which applies when comparing two security label components of type set for a read operation
3. DB2LBACREADTREE which applies when comparing two security label component of type tree for a read operation

Similarly, the write access rules are divided into three categories:

1. DB2LBACWRITEARRAY which applies when comparing two security label components of type array for a write operation
2. DB2LBACWRITESET which applies when comparing two security label components of type set for a write operation
3. DB2LBACWRITETREE which applies when comparing two security label component of type tree for a write operation


IDUG® 2007 North America

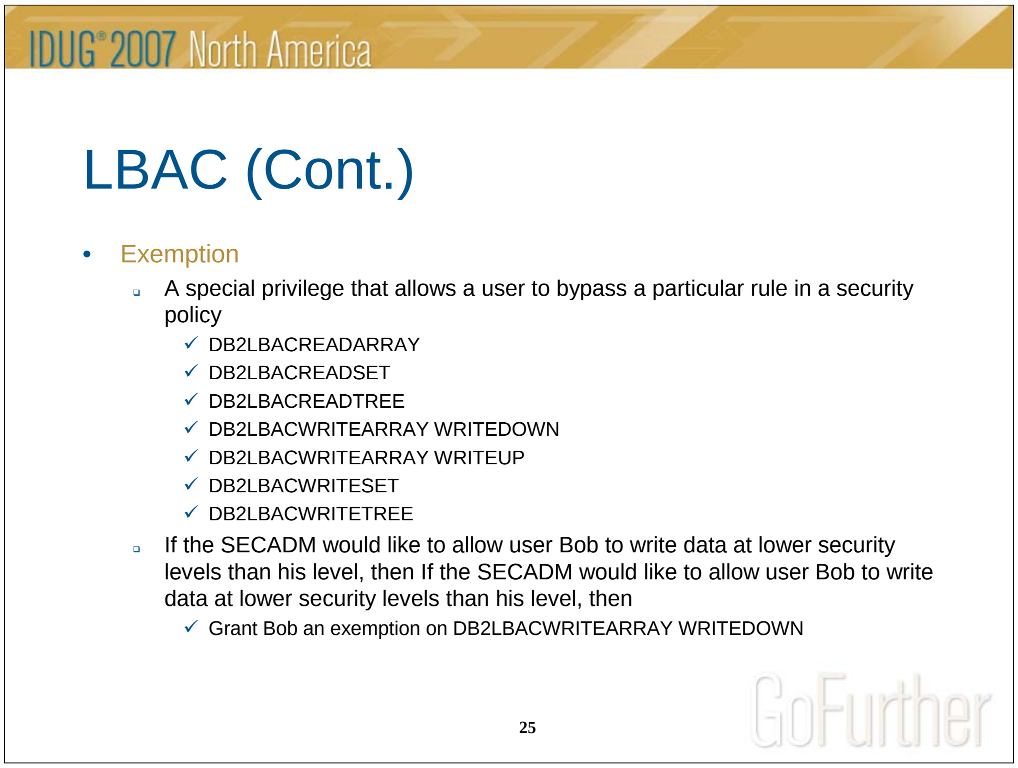
LBAC (Cont.)

- Security label
  - DB2 distinguishes between 3 types of security label
    - ✓ User Security Label: A security label that is granted to a database user
    - ✓ Row Security Label: A security label associated with a data row of a database table
    - ✓ Column Security Label: A security label associated with a column of a database table
  - A user can be granted a security label for
    - ✓ Read/write access only
    - ✓ Both read and write access
  - If the SECADM would like user Bob to read data with level 'S' and compartments {'A', 'B', 'C'}, but only write data with compartments {'C'}, then
    - ✓ Grant Bob a security label L1, where level = 'S' and compartments = {'A', 'B', 'C'} for read access
    - ✓ Grant Bob a security label L2, where level = 'S' and compartments = 'C' for write access

24

GoFurther

LBAC relies on security labels to perform access control. Security labels can be assigned to users, data rows and to columns. When assigning a security label to a user, the security administrator has the flexibility to grant that security label for read access only, for write access only or for both. For example, if a user’s responsibility is to just generate reports from a protected table, then, they need to be granted a security label for read access only. Similarly, if a user’s responsibility is to just load data into a protected table, then, they need to be granted a security label for write access only.


IDUG® 2007 North America

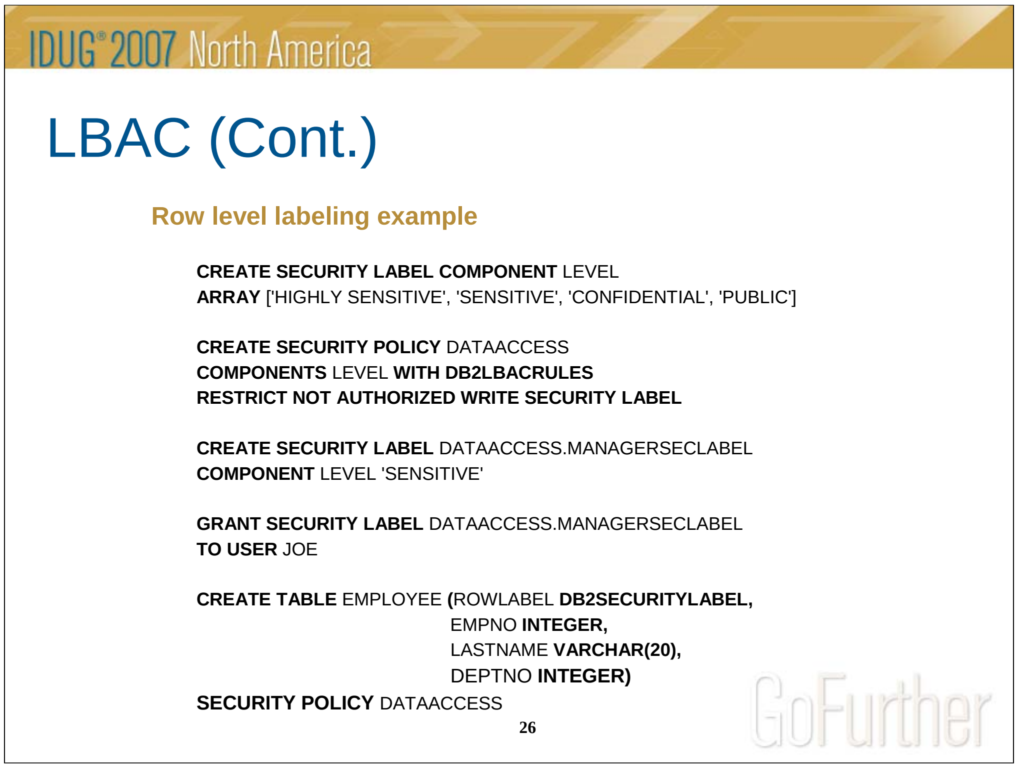
LBAC (Cont.)

- Exemption
  - A special privilege that allows a user to bypass a particular rule in a security policy
    - ✓ DB2LBACREADARRAY
    - ✓ DB2LBACREADSET
    - ✓ DB2LBACREADTREE
    - ✓ DB2LBACWRITEARRAY WRITEDOWN
    - ✓ DB2LBACWRITEARRAY WRITEUP
    - ✓ DB2LBACWRITESET
    - ✓ DB2LBACWRITETREE
  - If the SECADM would like to allow user Bob to write data at lower security levels than his level, then If the SECADM would like to allow user Bob to write data at lower security levels than his level, then
    - ✓ Grant Bob an exemption on DB2LBACWRITEARRAY WRITEDOWN

25

GoFurther

An exemption is a privilege to bypass a DB2 LBAC rule and which can be granted to a user. For example, if a user needs to write information to an array level lower than theirs, then, they need to be granted an exemption on the DB2LBACWRITEARRAY for WRITEDOWN. Otherwise, they can only write data at their own array level.

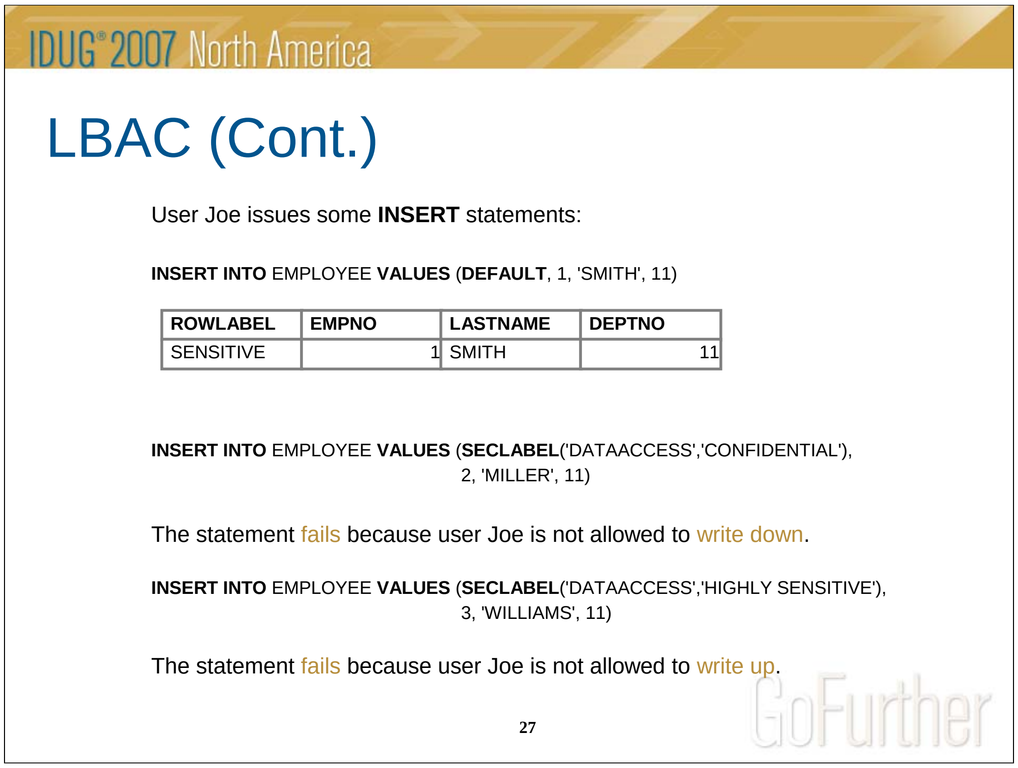


LBAC (Cont.)

Row level labeling example

```
CREATE SECURITY LABEL COMPONENT LEVEL
ARRAY ['HIGHLY SENSITIVE', 'SENSITIVE', 'CONFIDENTIAL', 'PUBLIC']

CREATE SECURITY POLICY DATAACCESS
COMPONENTS LEVEL WITH DB2LBACRULES
RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

CREATE SECURITY LABEL DATAACCESS.MANAGERSECLABEL
COMPONENT LEVEL 'SENSITIVE'

GRANT SECURITY LABEL DATAACCESS.MANAGERSECLABEL
TO USER JOE

CREATE TABLE EMPLOYEE (ROWLABEL DB2SECURITYLABEL,
                       EMPNO INTEGER,
                       LASTNAME VARCHAR(20),
                       DEPTNO INTEGER)
SECURITY POLICY DATAACCESS
```

The following example shows the SQL needed to set up a security policy and create a table protected with that security policy. The example also shows how you can create a security label and grant that security label to some database user, Joe in this example.


IDUG® 2007 North America

LBAC (Cont.)

User Joe issues some **INSERT** statements:

INSERT INTO EMPLOYEE **VALUES** (**DEFAULT**, 1, 'SMITH', 11)

ROWLABEL	EMPNO	LASTNAME	DEPTNO
SENSITIVE	1	SMITH	11

INSERT INTO EMPLOYEE **VALUES** (**SECLABEL**('DATAACCESS','CONFIDENTIAL'),
2, 'MILLER', 11)

The statement fails because user Joe is not allowed to write down.

INSERT INTO EMPLOYEE **VALUES** (**SECLABEL**('DATAACCESS','HIGHLY SENSITIVE'),
3, 'WILLIAMS', 11)

The statement fails because user Joe is not allowed to write up.

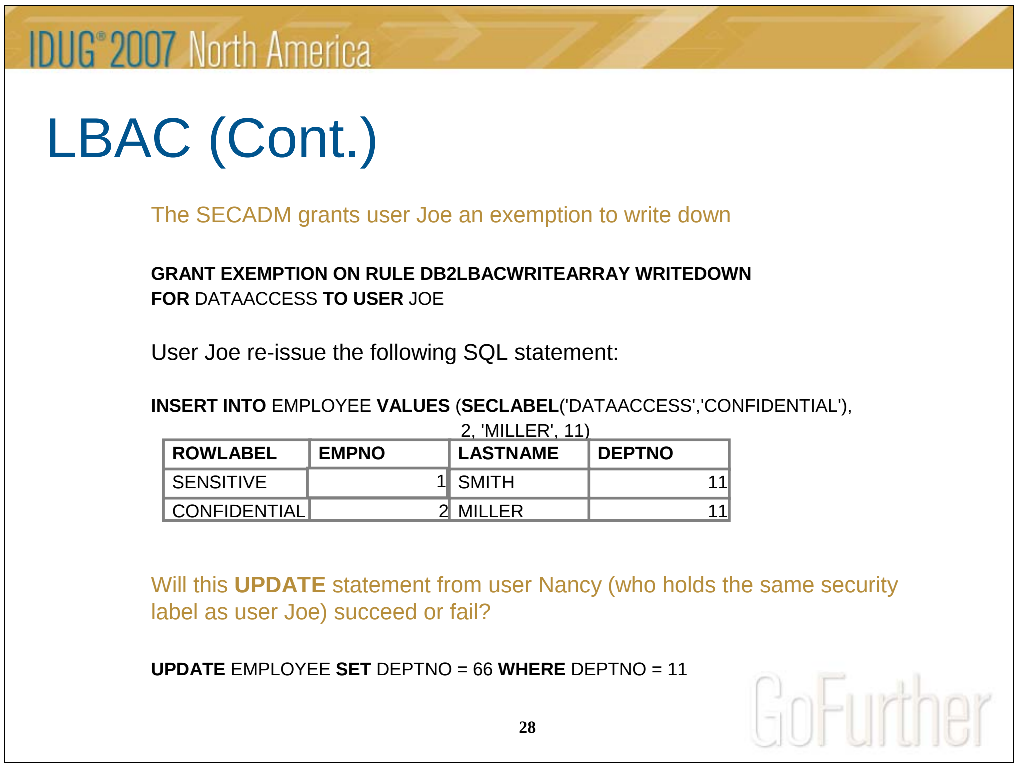
27

GoFurther

Joe attempts three INSERT SQL statements.

The first SQL statement succeeds because Joe’s own security label is used to label the data row inserted, and one is allowed to write data with their own security label (i.e., the DB2 LBAC WRITE rules are satisfied).

The second and third SQL statements fails because Joe is attempting a write down and a write up respectively, and he does not hold an exemption to do so.


IDUG® 2007 North America

LBAC (Cont.)

The SECADM grants user Joe an exemption to write down

GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
FOR DATAACCESS **TO USER** JOE

User Joe re-issue the following SQL statement:

INSERT INTO EMPLOYEE **VALUES** (**SECLABEL**('DATAACCESS','CONFIDENTIAL'),
2, 'MILLER', 11)

ROWLABEL	EMPNO	LASTNAME	DEPTNO
SENSITIVE	1	SMITH	11
CONFIDENTIAL	2	MILLER	11

Will this **UPDATE** statement from user Nancy (who holds the same security label as user Joe) succeed or fail?

UPDATE EMPLOYEE **SET** DEPTNO = 66 **WHERE** DEPTNO = 11

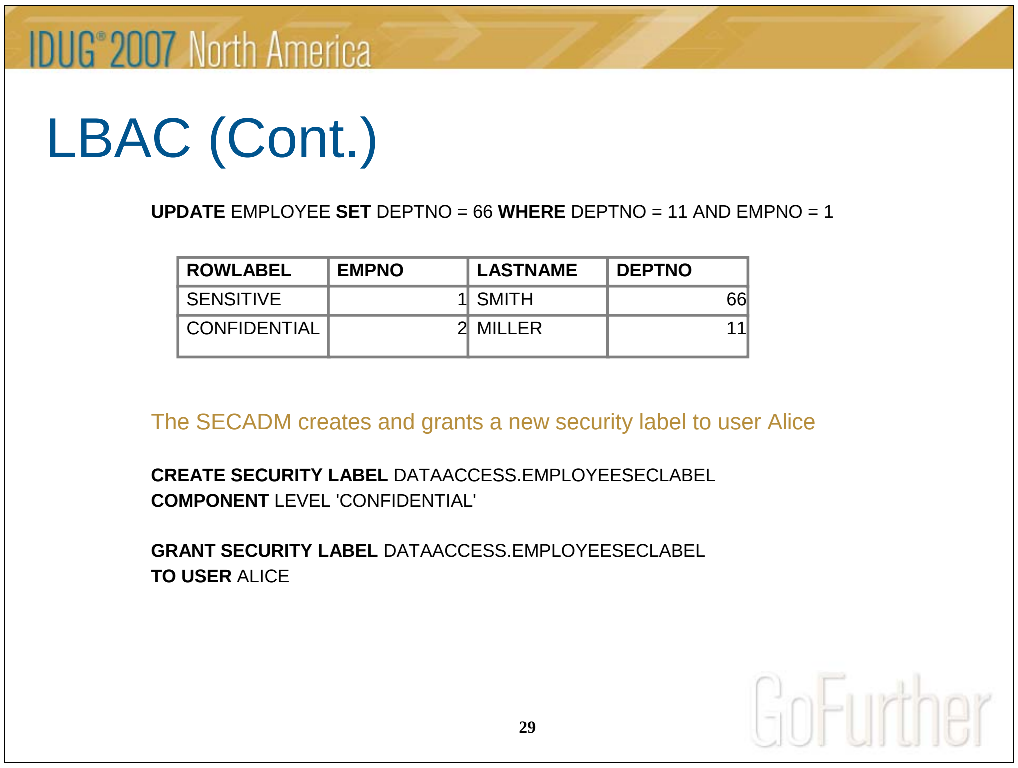
28

GoFurther

The security administrator decides to grant Joe an exemption to write down.

Now, Joe issues the SQL statement that previously failed because of write down violation. This time the statement succeeds because Joe holds an exemption to write down.

The UPDATE statement by user Nancy fails because it affects 2 rows, one for which she is allowed write access and one for which she is not allowed write access (the CONFIDENTIAL row).



IDUG® 2007 North America

LBAC (Cont.)

```
UPDATE EMPLOYEE SET DEPTNO = 66 WHERE DEPTNO = 11 AND EMPNO = 1
```

ROWLABEL	EMPNO	LASTNAME	DEPTNO
SENSITIVE	1	SMITH	66
CONFIDENTIAL	2	MILLER	11

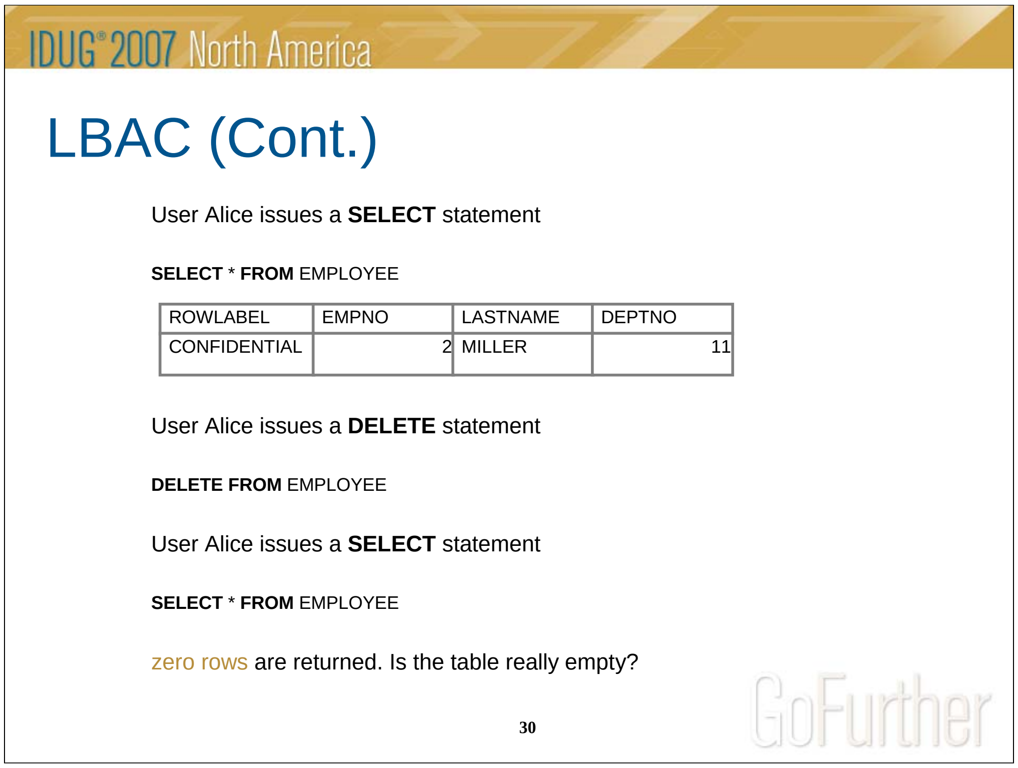
The SECADM creates and grants a new security label to user Alice

```
CREATE SECURITY LABEL DATAACCESS.EMPLOYEESECLABEL
COMPONENT LEVEL 'CONFIDENTIAL'

GRANT SECURITY LABEL DATAACCESS.EMPLOYEESECLABEL
TO USER ALICE
```

29

GoFurther

The modified UPDATE statement succeeds because it now affects a single row (the SENSITIVE row) and Nancy is allowed write access for SENSITIVE rows.



IDUG® 2007 North America

LBAC (Cont.)

User Alice issues a **SELECT** statement

SELECT * **FROM** EMPLOYEE

ROWLABEL	EMPNO	LASTNAME	DEPTNO
CONFIDENTIAL	2	MILLER	11

User Alice issues a **DELETE** statement

DELETE FROM EMPLOYEE

User Alice issues a **SELECT** statement

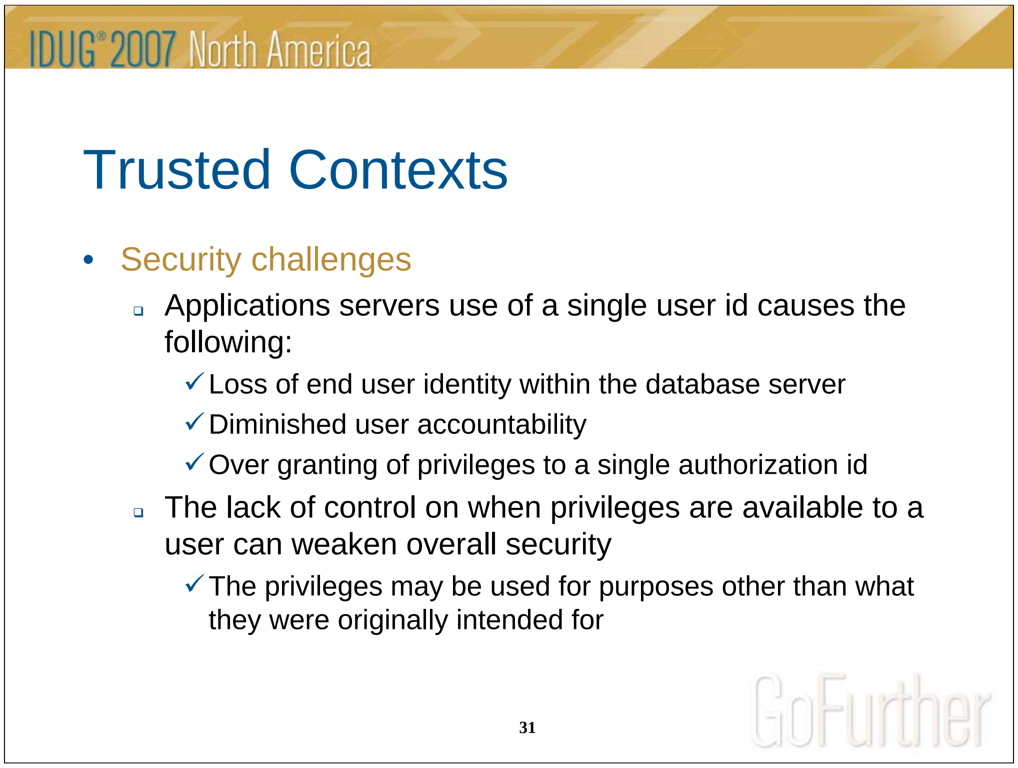
SELECT * **FROM** EMPLOYEE

zero rows are returned. Is the table really empty?

30

GoFurther

The table is not really empty because there exists another row at the SENSITIVE level and for which Alice does not have access. In other words, the row does not exist as far as Alice is concerned.



IDUG® 2007 North America

Trusted Contexts

- Security challenges
  - Applications servers use of a single user id causes the following:
    - ✓ Loss of end user identity within the database server
    - ✓ Diminished user accountability
    - ✓ Over granting of privileges to a single authorization id
  - The lack of control on when privileges are available to a user can weaken overall security
    - ✓ The privileges may be used for purposes other than what they were originally intended for

31

GoFurther

The three-tiered application model extends the standard two-tiered client and server model by placing a middle tier between the client application and the database server. It has gained great popularity in the recent years particularly with the emergence of web-based technologies and the Java 2 Enterprise Edition (J2EE) platform. Some of the software products that support the three-tier application model are IBM's Websphere Application Server (WAS), Domino, and the .NET platform used by SAP R/3.

In a three-tiered application model, the middle tier is responsible for, authenticating the users running the client applications and for managing the interactions with the database server. Traditionally, all the interactions with the database server occur through a database connection established by the middle tier using a combination of a user ID and a credential that identify that middle tier to the database server. In other words, the database server uses the database privileges associated with the middle tier's user ID for all authorization checking and auditing that must occur for any database access, including those accesses performed by the middle tier on behalf of a user.

While the three-tiered application model has many benefits, having all interactions with the database server (e.g., a user request) occur under the middle tier's authorization ID raises several security concerns, which can be summarized as follows:

1. Loss of user identity: Some enterprises prefer to know the identity of the actual user accessing the database for access control purposes.

2. Diminished user accountability: Accountability through auditing is a basic principle in database security. Not knowing the user's identity makes it difficult to distinguish the transactions performed by the middle tier for its own purpose from those performed by the middle tier on behalf of some user.

3. Over granting of privileges to the middle tier's authorization ID: The middle tier's authorization ID must have all the privileges needed to execute all the requests from all the users. This has the security issue of enabling users who do not need access to certain information to obtain access to them.

4. Weakened security: In addition to the privilege issue raised in the previous point, the current approach requires that the authorization ID used by the middle tier to connect must be granted privileges on all resources that might be accessed by user requests. If that middle-tier authorization ID is ever compromised, then all those resources will be exposed.

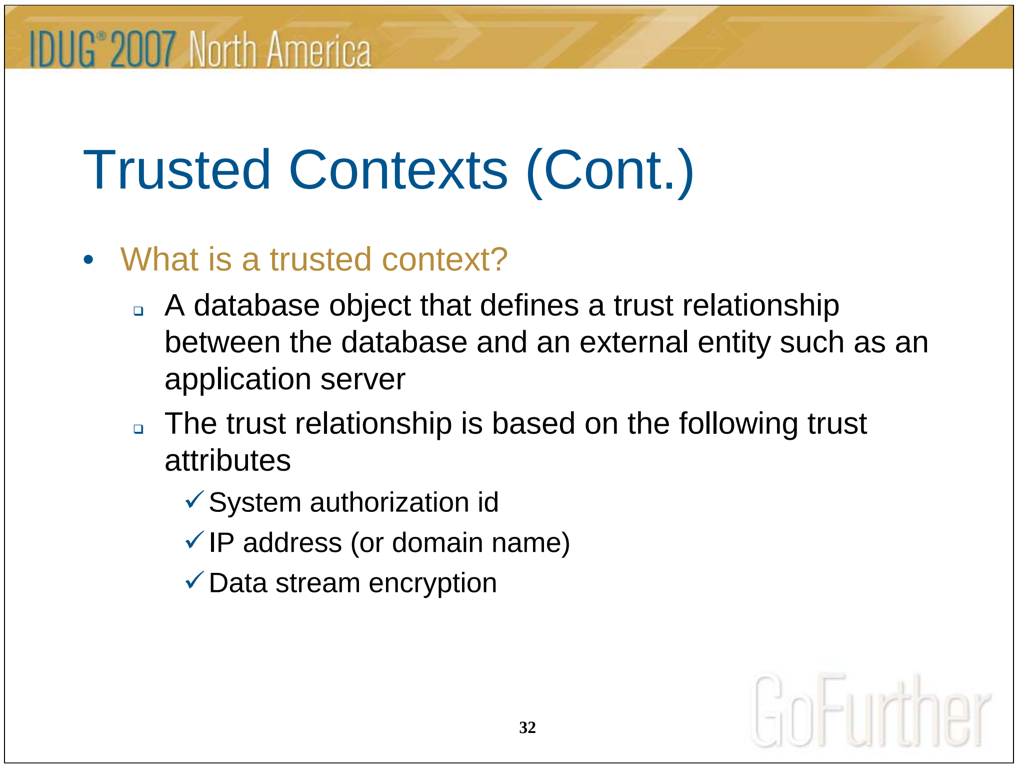
Clearly, there is a need for a mechanism whereby the actual user's identity and database privileges are used for database requests performed by the middle tier on behalf of that user. The most straightforward approach of achieving this goal would be for the middle-tier to establish a new connection using the user's ID and password, and then direct the user's requests through that connection. Although simple, this approach suffers from several drawbacks which include the following:

1. Inapplicability for certain middle tiers: Many middle-tier servers do not have the user authentication credentials needed to establish a connection.

2. Performance overhead: There is an obvious performance overhead associated with creating a new physical connection and re-authenticating the user at the database server.

3. Maintenance overhead: There is a maintenance overhead of having two user definitions (one on the middle tier and one at the server). This requires changing passwords at different places.

The DB2 Trusted Contexts capability addresses the problem above. It also offers an additional advantage which is the ability to control when a privilege is made available to a database user. The lack of control on when privileges are available to a user can weaken overall security. For example, privileges may be used for purposes other than what they were originally intended for.


IDUG® 2007 North America

Trusted Contexts (Cont.)

- What is a trusted context?
  - A database object that defines a trust relationship between the database and an external entity such as an application server
  - The trust relationship is based on the following trust attributes
    - ✓ System authorization id
    - ✓ IP address (or domain name)
    - ✓ Data stream encryption

32

GoFurther

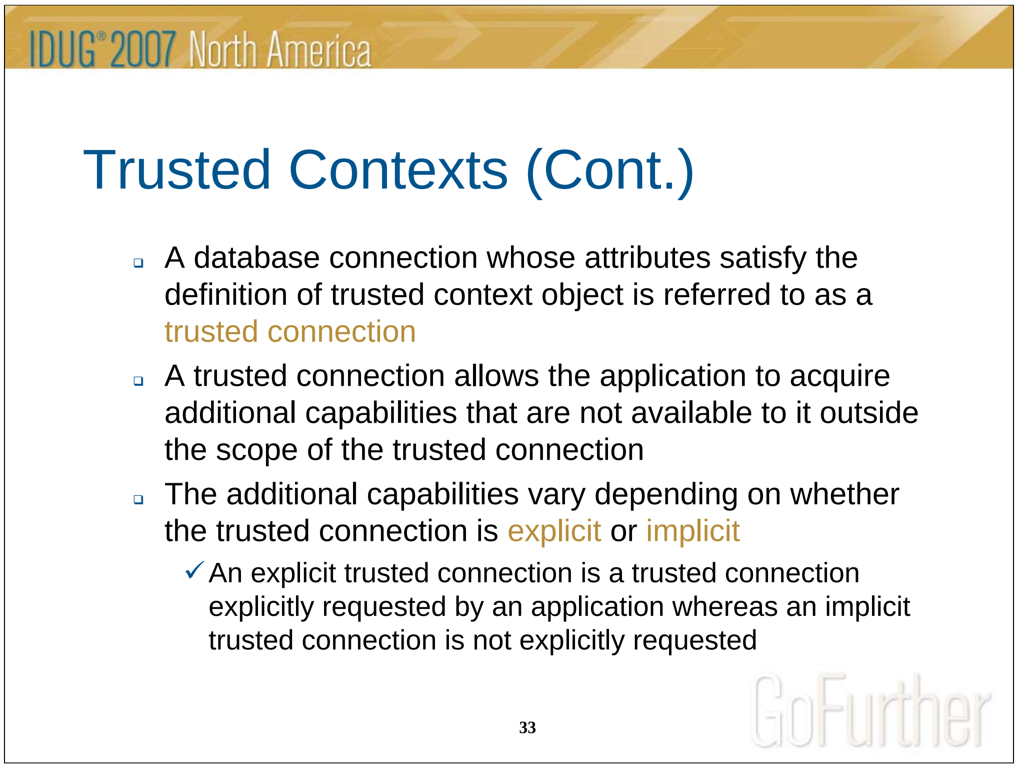
A trusted context is a database object that defines a trust relationship between the database server and an external entity such as an application server.

The trust relationship is based on the following trust attributes which are the following:

1. System authorization id: This is the authorization ID that establishes a database connection.

2. IP address (or domain name): This is the IP address from which the database connection originates

3. Data stream encryption: This is the encryption setting for the data communication between the database server and the client



IDUG® 2007 North America

Trusted Contexts (Cont.)

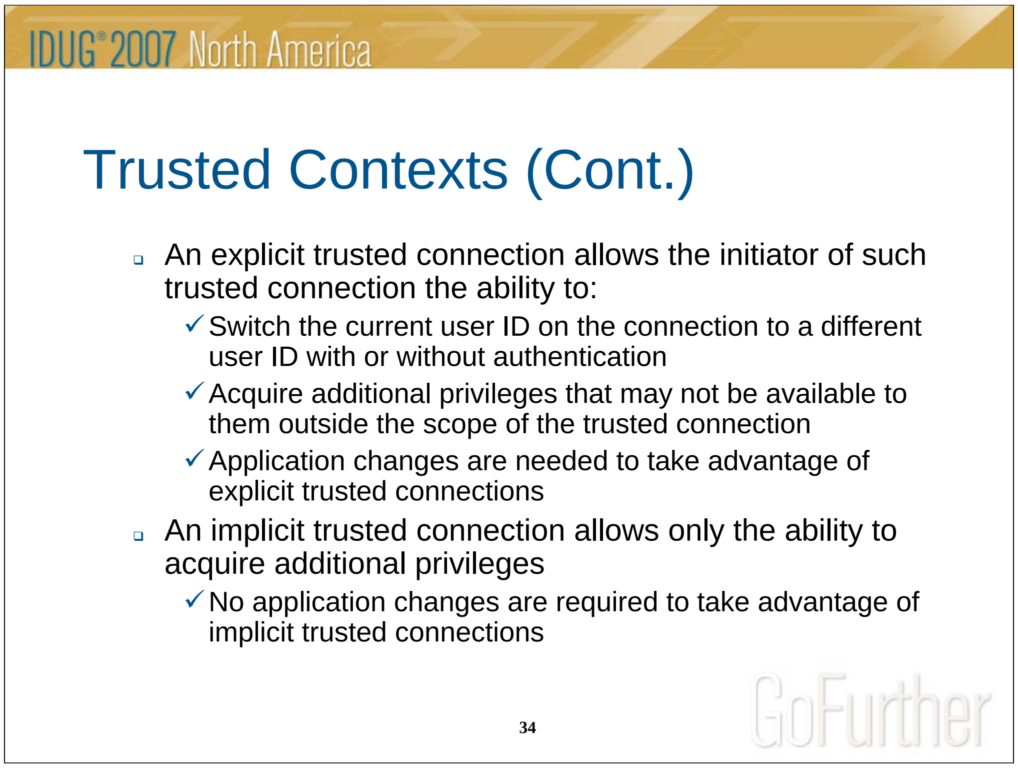
- A database connection whose attributes satisfy the definition of trusted context object is referred to as a trusted connection
- A trusted connection allows the application to acquire additional capabilities that are not available to it outside the scope of the trusted connection
- The additional capabilities vary depending on whether the trusted connection is explicit or implicit
  - ✓ An explicit trusted connection is a trusted connection explicitly requested by an application whereas an implicit trusted connection is not explicitly requested

33

GoFurther

A trusted connection is a database connection whose attributes satisfy the definition of trusted context object.

When established, a trusted connection allows the authorization ID of the connection to acquire additional capabilities that are not available to it outside the scope of the trusted connection. These capabilities vary depending on whether the trusted connection is explicit or implicit (an explicit trusted connection is a trusted connection explicitly requested by an application whereas an implicit trusted connection is not explicitly requested).



IDUG® 2007 North America

Trusted Contexts (Cont.)

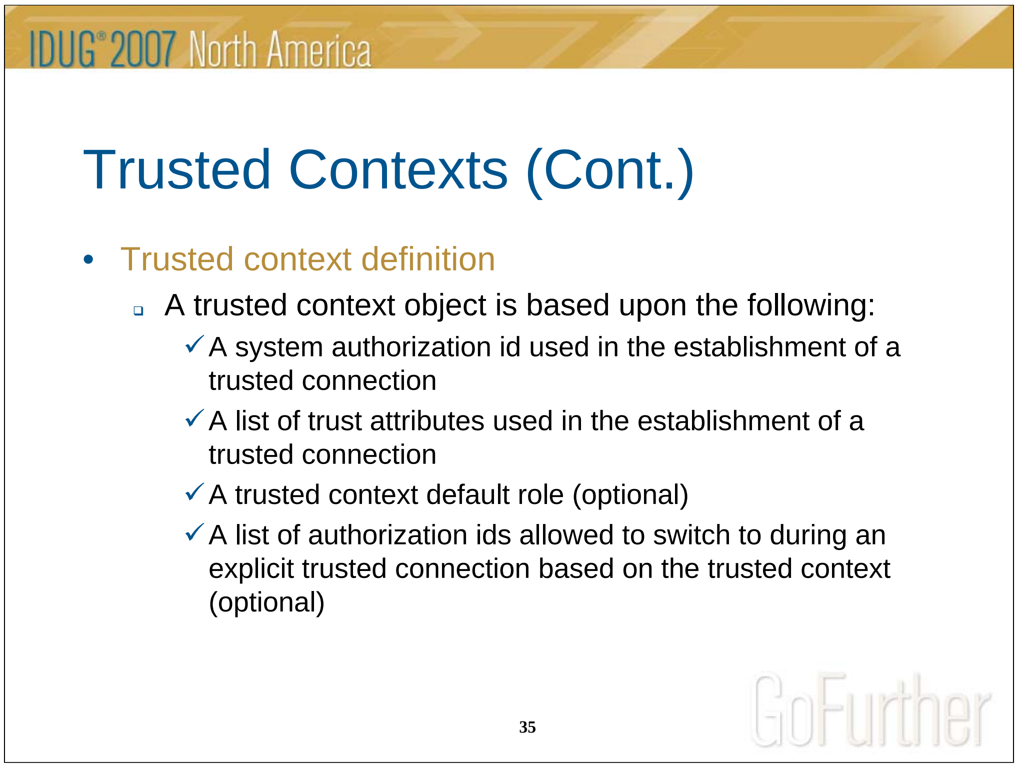
- An explicit trusted connection allows the initiator of such trusted connection the ability to:
  - ✓ Switch the current user ID on the connection to a different user ID with or without authentication
  - ✓ Acquire additional privileges that may not be available to them outside the scope of the trusted connection
  - ✓ Application changes are needed to take advantage of explicit trusted connections
- An implicit trusted connection allows only the ability to acquire additional privileges
  - ✓ No application changes are required to take advantage of implicit trusted connections

34

GoFurther

An explicit trusted connection allows the authorization ID of the connection to (1) switch the current user ID on the connection to a different user ID with or without authentication, and (2) acquire additional privileges that may not be available to them outside the scope of the trusted connection.

Implicit trusted connections only allow the ability to acquire additional privileges.


IDUG® 2007 North America

Trusted Contexts (Cont.)

- Trusted context definition
  - A trusted context object is based upon the following:
    - ✓ A system authorization id used in the establishment of a trusted connection
    - ✓ A list of trust attributes used in the establishment of a trusted connection
    - ✓ A trusted context default role (optional)
    - ✓ A list of authorization ids allowed to switch to during an explicit trusted connection based on the trusted context (optional)

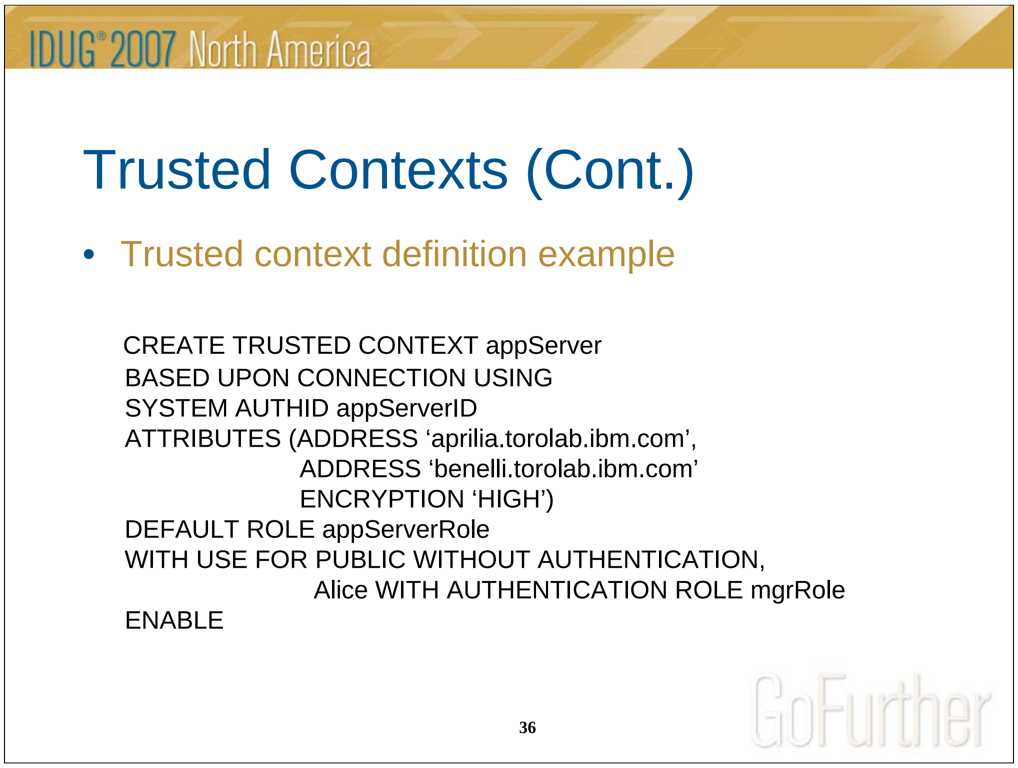
35

GoFurther

A trusted context is a database object that can be created using the CREATE TRUSTED CONTEXT SQL statement.

There are four main parts in the definition of a trusted context object:

1. A system authorization id used in the establishment of a trusted connection
2. A list of trust attributes used in the establishment of a trusted connection
3. A trusted context default role (optional)
4. A list of authorization ids allowed to switch to during an explicit trusted connection based on the trusted context (optional)


IDUG® 2007 North America

Trusted Contexts (Cont.)

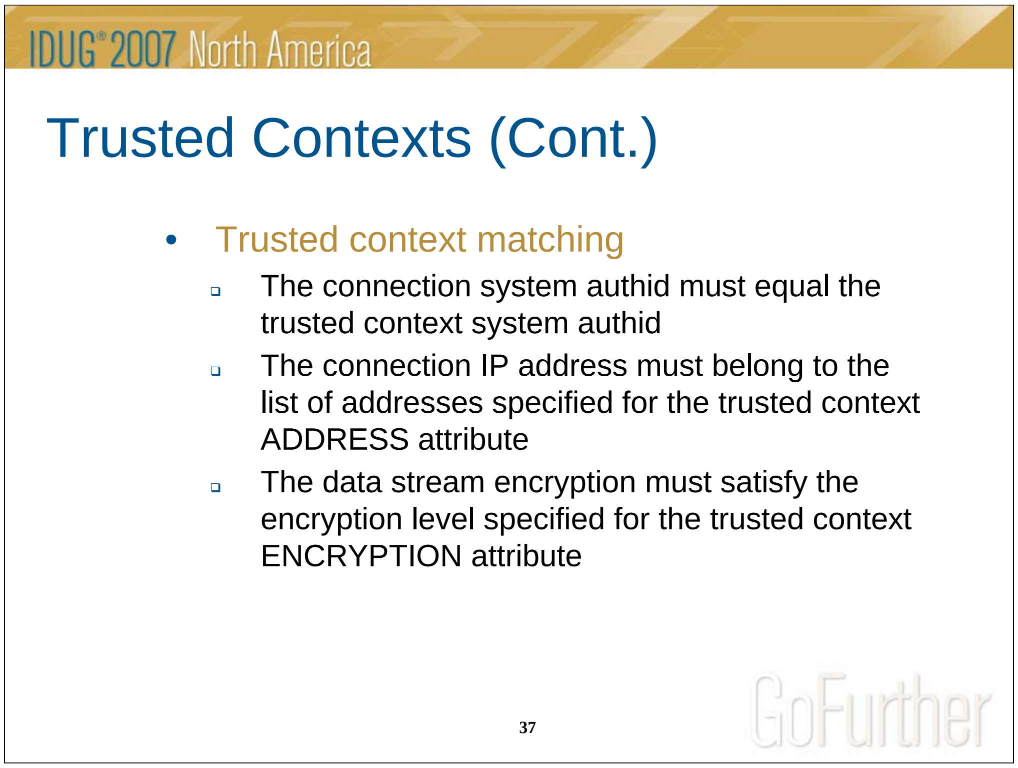
- Trusted context definition example

```
CREATE TRUSTED CONTEXT appServer
BASED UPON CONNECTION USING
SYSTEM AUTHID appServerID
ATTRIBUTES (ADDRESS ‘aprilia.torolab.ibm.com’,
                  ADDRESS ‘benelli.torolab.ibm.com’
                  ENCRYPTION ‘HIGH’)
DEFAULT ROLE appServerRole
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION,
                   Alice WITH AUTHENTICATION ROLE mgrRole
ENABLE
```

36

GoFurther

This is an example CREATE TRUSTED CONTEXT SQL statement.



IDUG® 2007 North America

Trusted Contexts (Cont.)

- Trusted context matching
  - The connection system authid must equal the trusted context system authid
  - The connection IP address must belong to the list of addresses specified for the trusted context ADDRESS attribute
  - The data stream encryption must satisfy the encryption level specified for the trusted context ENCRYPTION attribute

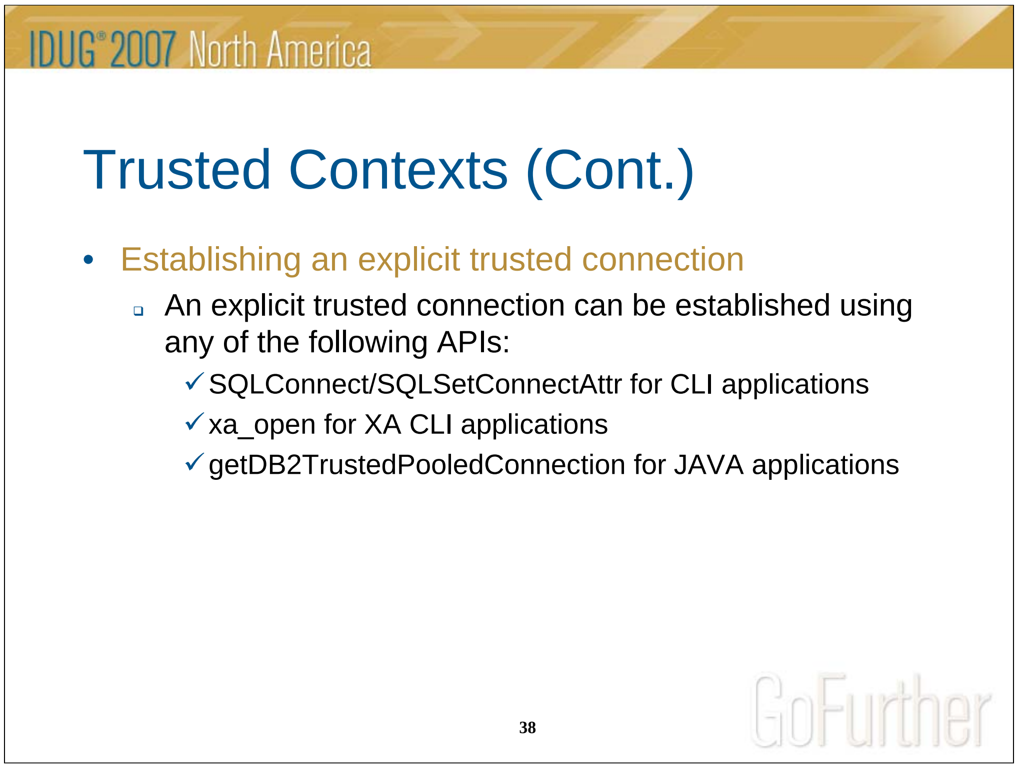
37

GoFurther

Each incoming connection is evaluated for trust with respect to a trusted context object definition in the database.

In order for a database connection to be trusted, the following must be true:

1. The authorization ID of the connection matches the system authorization ID specified for a trusted context object
2. The IP address from which the connection was initiated belongs to the set of IP addresses specified for that trusted context object. If the trusted context object did not specify any IP address, then, this condition is satisfied regardless of where the connection was initiated from
3. The encryption setting for the data communication between the database server and the database client matches the encryption setting for that trusted context object (see table above for further details). If the trusted context definition did not specify any encryption setting, then, this condition is satisfied regardless of the encryption setting for the data communication between the database server and the database client


IDUG® 2007 North America

Trusted Contexts (Cont.)

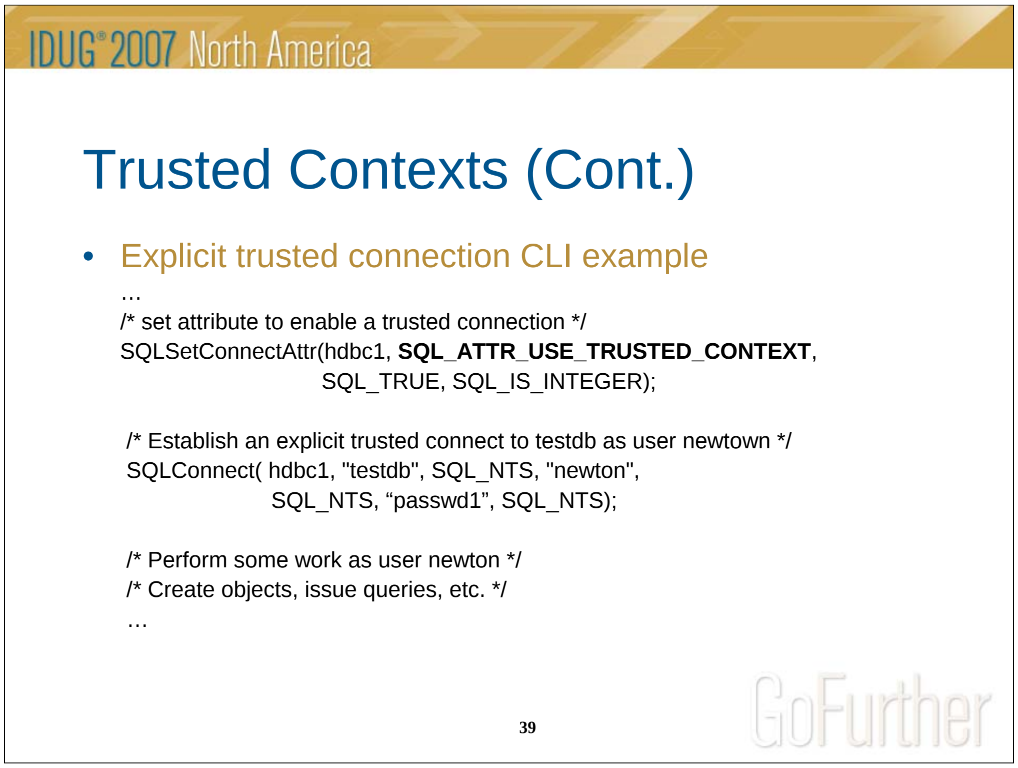
- Establishing an explicit trusted connection
  - An explicit trusted connection can be established using any of the following APIs:
    - ✓ SQLConnect/SQLSetConnectAttr for CLI applications
    - ✓ xa_open for XA CLI applications
    - ✓ getDB2TrustedPooledConnection for JAVA applications

38

GoFurther

There are no specific requirements on how to establish an implicit trusted connection. However, an explicit trusted connection must be established using one of the following:

1. SQLConnect/SQLSetConnectAttr for CLI applications
2. xa_open for XA CLI applications
3. getDB2TrustedPooledConnection for JAVA applications


IDUG® 2007 North America

Trusted Contexts (Cont.)

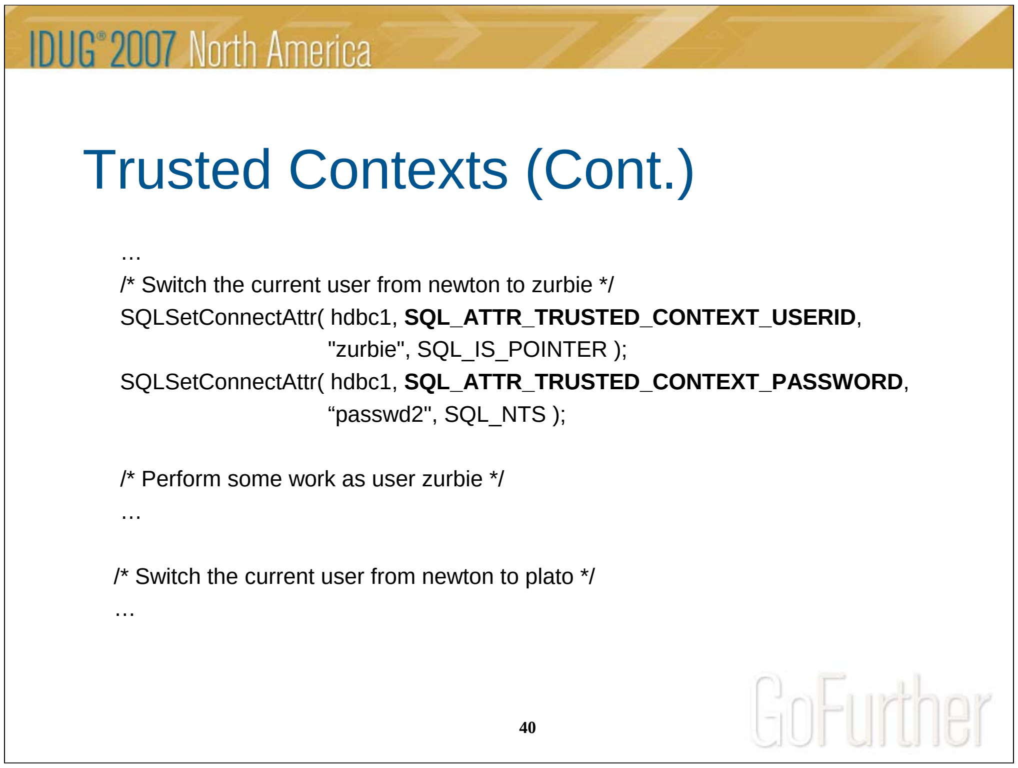
- Explicit trusted connection CLI example

```
…
/* set attribute to enable a trusted connection */
SQLSetConnectAttr(hdbc1, SQL_ATTR_USE_TRUSTED_CONTEXT,
                        SQL_TRUE, SQL_IS_INTEGER);

 /* Establish an explicit trusted connect to testdb as user newtown */
 SQLConnect( hdbc1, "testdb", SQL_NTS, "newton",
                   SQL_NTS, “passwd1”, SQL_NTS);

 /* Perform some work as user newton */
 /* Create objects, issue queries, etc. */
 …
```

39

GoFurther

This is a CLI example showing how an explicit trusted connection can be established using SQLSetConnectAttr and SQLConnect. Basically, you need to set the **SQL_ATTR_USE_TRUSTED_CONTEXT** attribute.



IDUG® 2007 North America

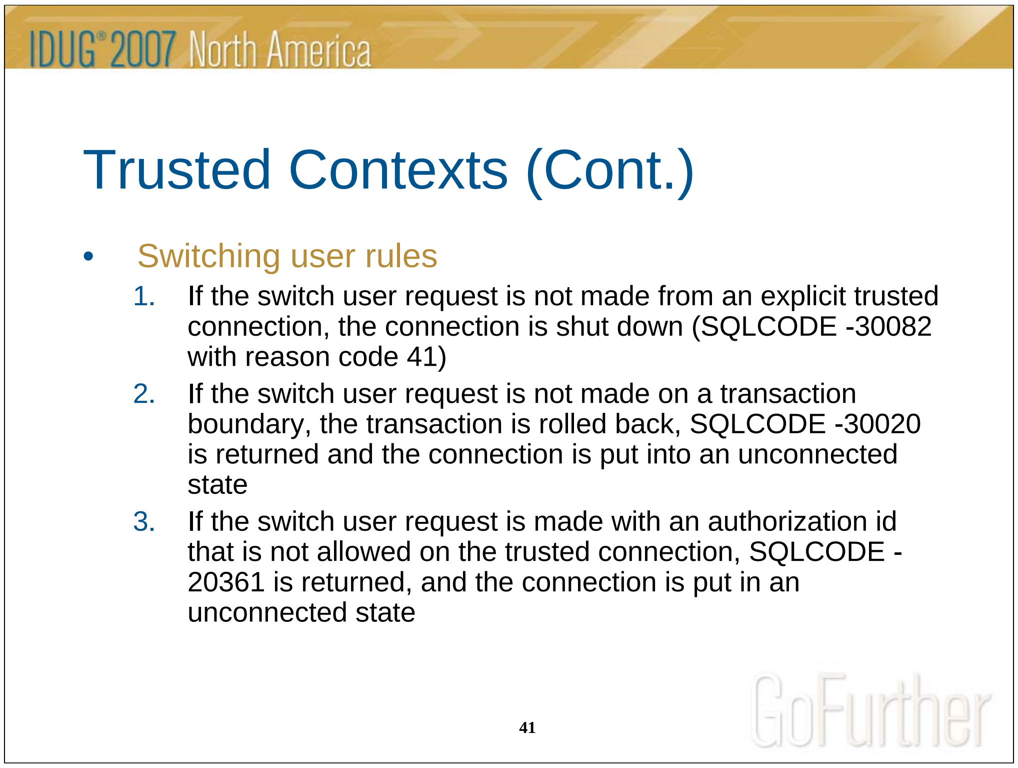
Trusted Contexts (Cont.)

```
…
/* Switch the current user from newton to zurbie */
SQLSetConnectAttr( hdbc1, SQL_ATTR_TRUSTED_CONTEXT_USERID,
                          "zurbie", SQL_IS_POINTER );
SQLSetConnectAttr( hdbc1, SQL_ATTR_TRUSTED_CONTEXT_PASSWORD,
                          “passwd2", SQL_NTS );

/* Perform some work as user zurbie */
…

/* Switch the current user from newton to plato */
…
```

40

GoFurther

The example also shows how to switch the current user ID on the explicit trusted connection using SQLSetConnectAttr. Basically, you need to set the **SQL_ATTR_TRUSTED_CONTEXT_USERID** and optionally the **SQL_ATTR_TRUSTED_CONTEXT_PASSWORD** if authentication is required.



IDUG® 2007 North America

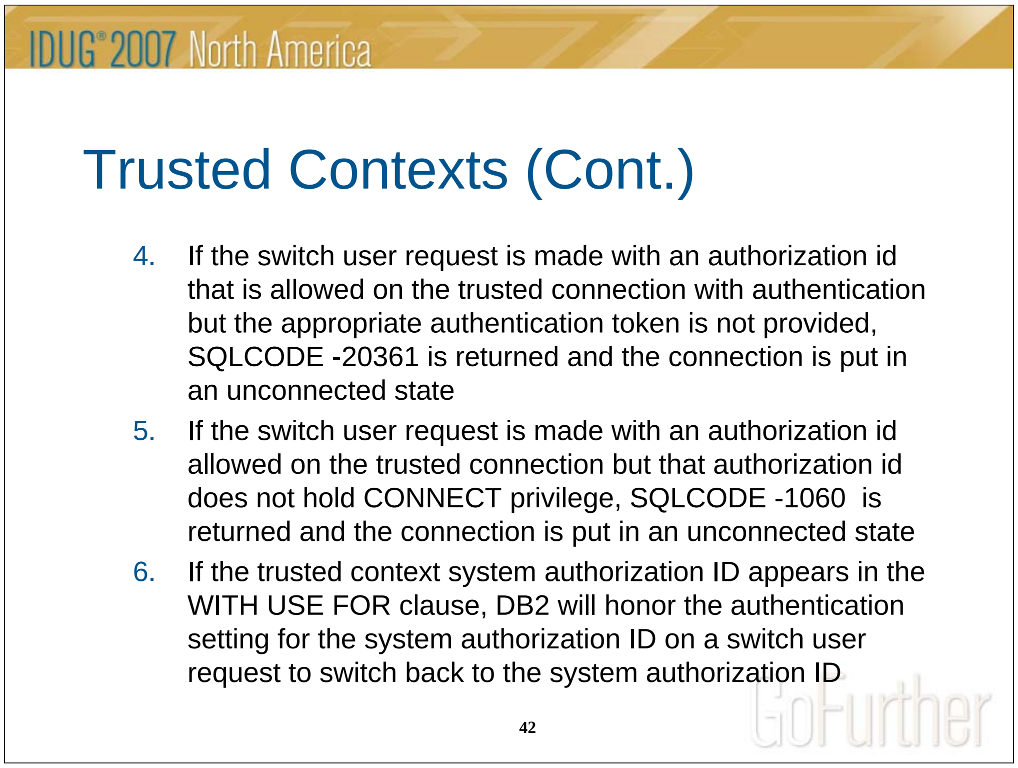
Trusted Contexts (Cont.)

- Switching user rules
  1. If the switch user request is not made from an explicit trusted connection, the connection is shut down (SQLCODE -30082 with reason code 41)
  2. If the switch user request is not made on a transaction boundary, the transaction is rolled back, SQLCODE -30020 is returned and the connection is put into an unconnected state
  3. If the switch user request is made with an authorization id that is not allowed on the trusted connection, SQLCODE -20361 is returned, and the connection is put in an unconnected state

41

GoFurther

There are a number of rules that govern switching the current user ID on an explicit trusted connection. These rules are listed on this chart and on the following 2 charts.


IDUG® 2007 North America

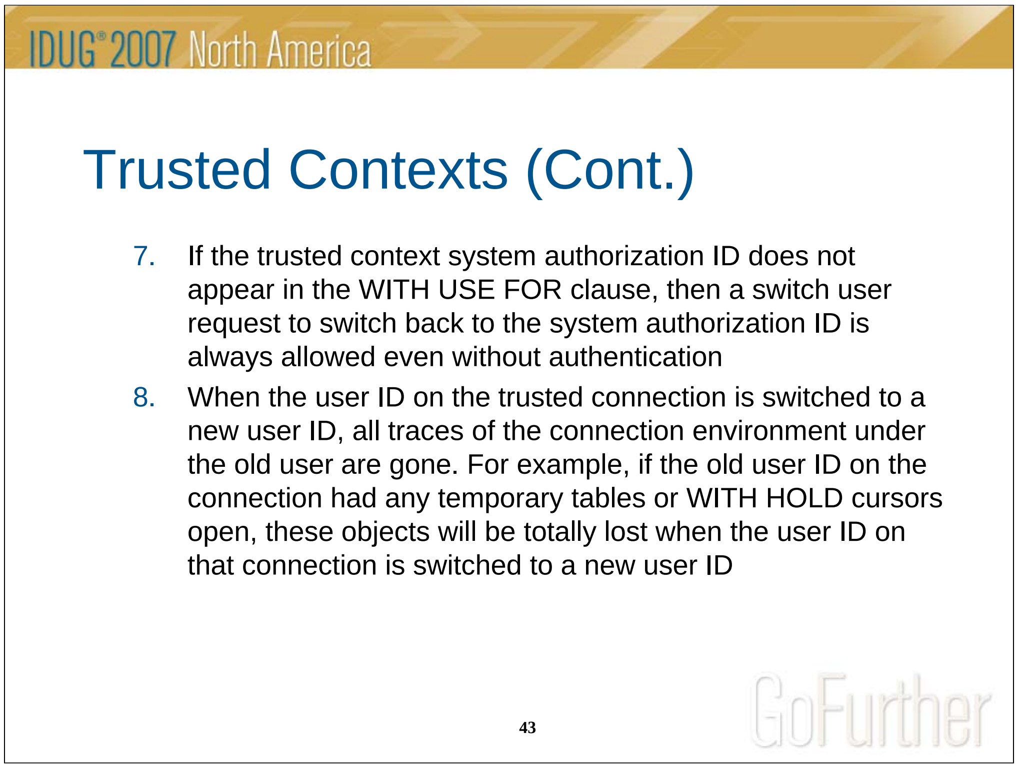
Trusted Contexts (Cont.)

4. If the switch user request is made with an authorization id that is allowed on the trusted connection with authentication but the appropriate authentication token is not provided, SQLCODE -20361 is returned and the connection is put in an unconnected state
5. If the switch user request is made with an authorization id allowed on the trusted connection but that authorization id does not hold CONNECT privilege, SQLCODE -1060 is returned and the connection is put in an unconnected state
6. If the trusted context system authorization ID appears in the WITH USE FOR clause, DB2 will honor the authentication setting for the system authorization ID on a switch user request to switch back to the system authorization ID

42

GoFurther

Switching user rules (continued).



IDUG® 2007 North America

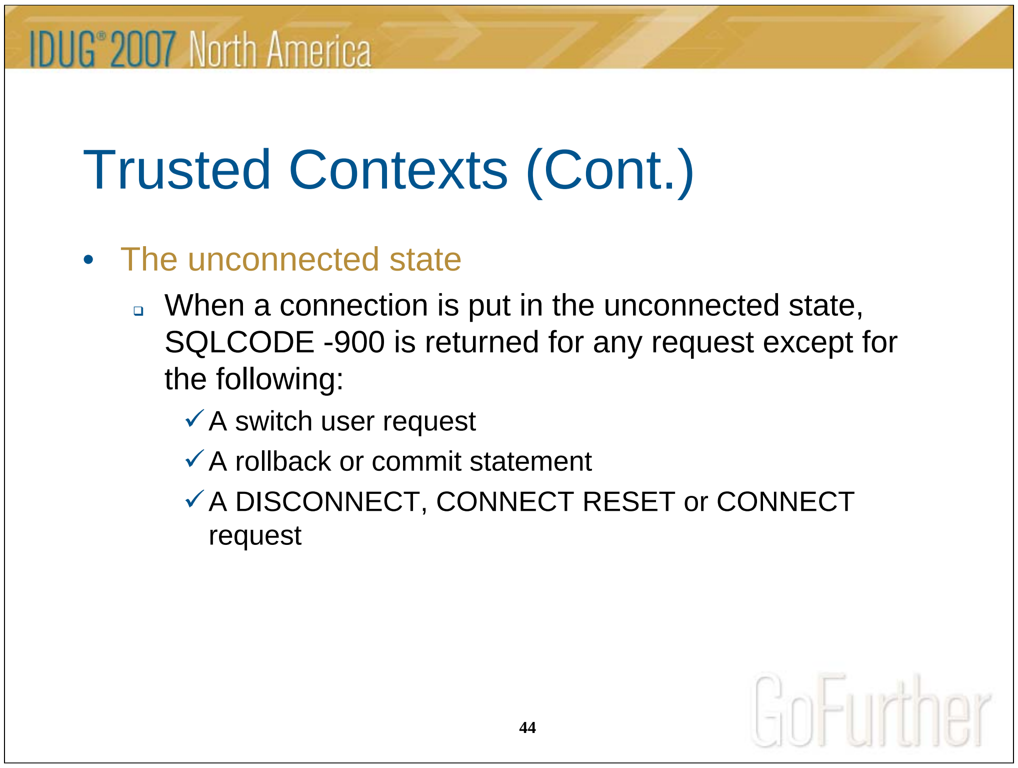
Trusted Contexts (Cont.)

7. If the trusted context system authorization ID does not appear in the WITH USE FOR clause, then a switch user request to switch back to the system authorization ID is always allowed even without authentication
8. When the user ID on the trusted connection is switched to a new user ID, all traces of the connection environment under the old user are gone. For example, if the old user ID on the connection had any temporary tables or WITH HOLD cursors open, these objects will be totally lost when the user ID on that connection is switched to a new user ID

43

GoFurther

Switching user rules (continued).


IDUG® 2007 North America

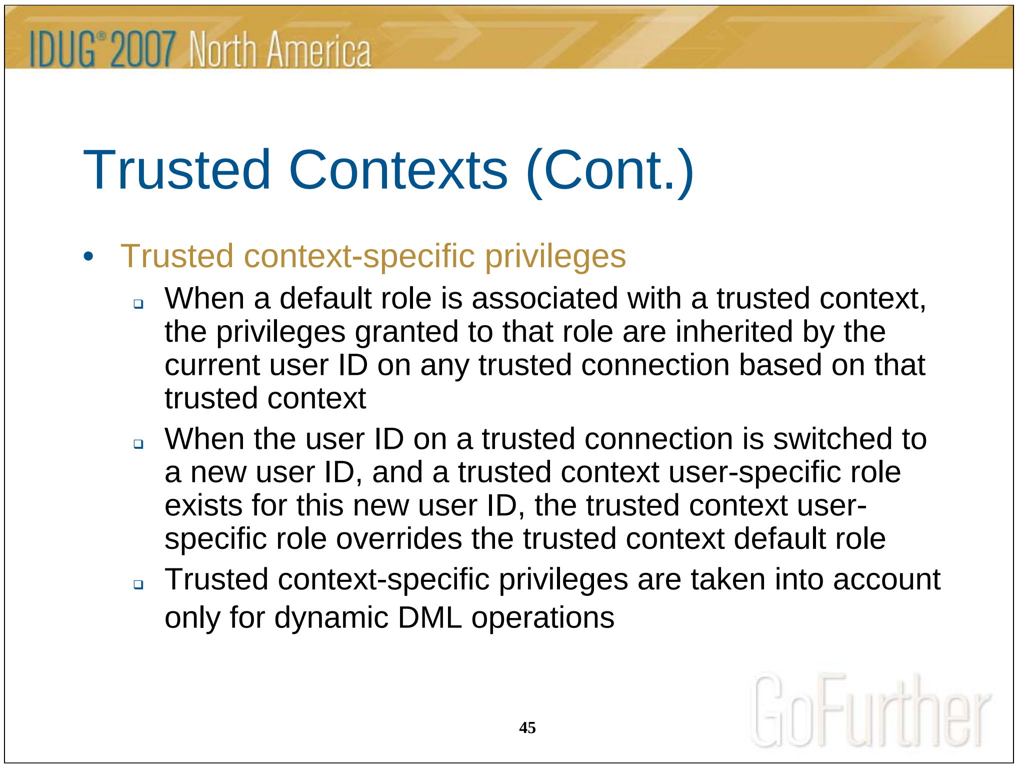
Trusted Contexts (Cont.)

- The unconnected state
  - When a connection is put in the unconnected state, SQLCODE -900 is returned for any request except for the following:
    - ✓ A switch user request
    - ✓ A rollback or commit statement
    - ✓ A DISCONNECT, CONNECT RESET or CONNECT request

44

GoFurther

Switching user rules (continued).



IDUG® 2007 North America

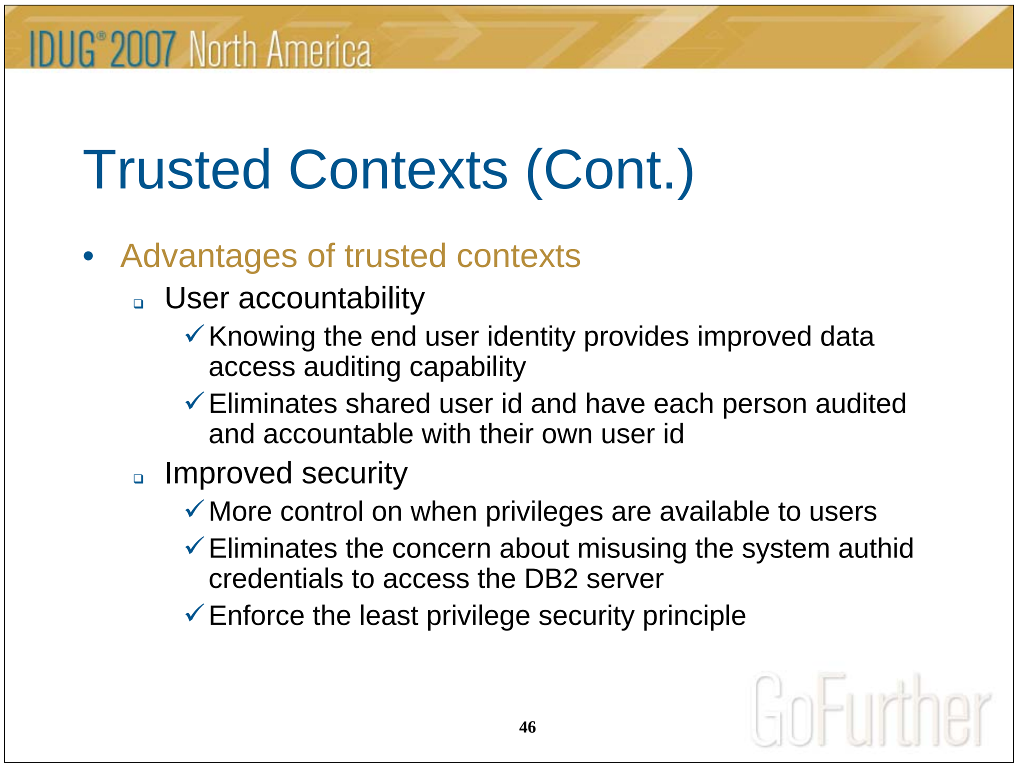
Trusted Contexts (Cont.)

- Trusted context-specific privileges
  - When a default role is associated with a trusted context, the privileges granted to that role are inherited by the current user ID on any trusted connection based on that trusted context
  - When the user ID on a trusted connection is switched to a new user ID, and a trusted context user-specific role exists for this new user ID, the trusted context user-specific role overrides the trusted context default role
  - Trusted context-specific privileges are taken into account only for dynamic DML operations

45

GoFurther

You can use trusted contexts to gain more control on when a privilege is available to a user. For example, you can grant the privilege to a database role and then assign the role to a trusted context object. A user can only take advantage of the role if they can establish a trusted connection based on that trusted context object.



IDUG® 2007 North America

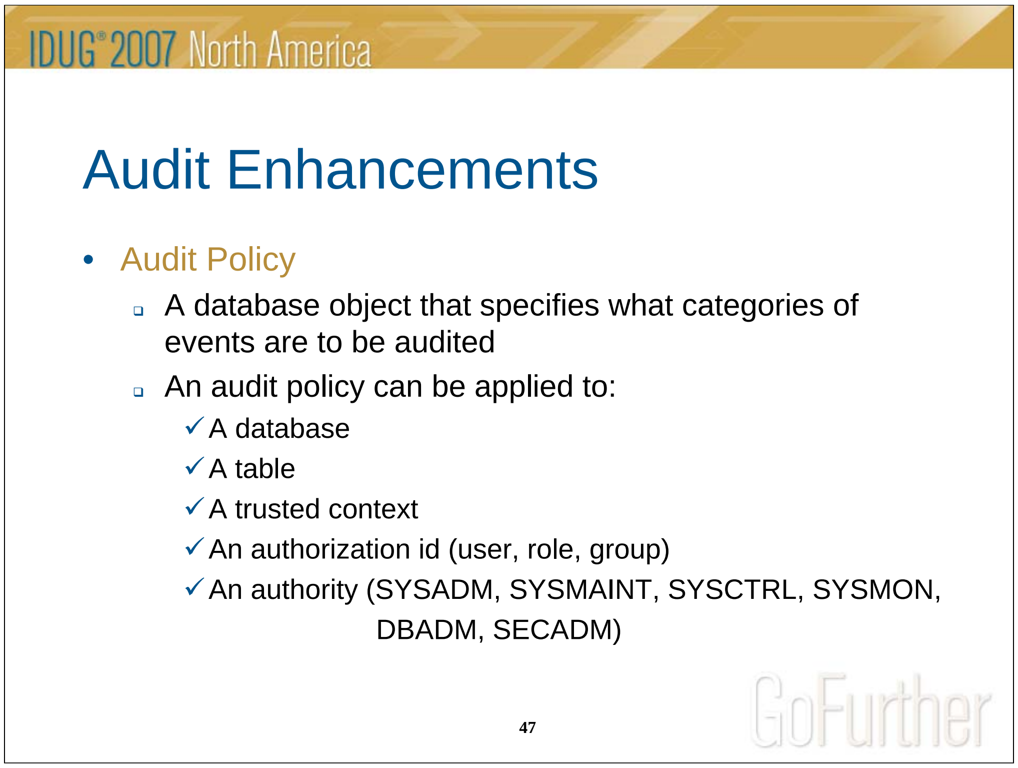
Trusted Contexts (Cont.)

- Advantages of trusted contexts
  - User accountability
    - ✓ Knowing the end user identity provides improved data access auditing capability
    - ✓ Eliminates shared user id and have each person audited and accountable with their own user id
  - Improved security
    - ✓ More control on when privileges are available to users
    - ✓ Eliminates the concern about misusing the system authid credentials to access the DB2 server
    - ✓ Enforce the least privilege security principle

46

GoFurther

This chart summarizes the advantages of the trusted contexts capability.



IDUG® 2007 North America

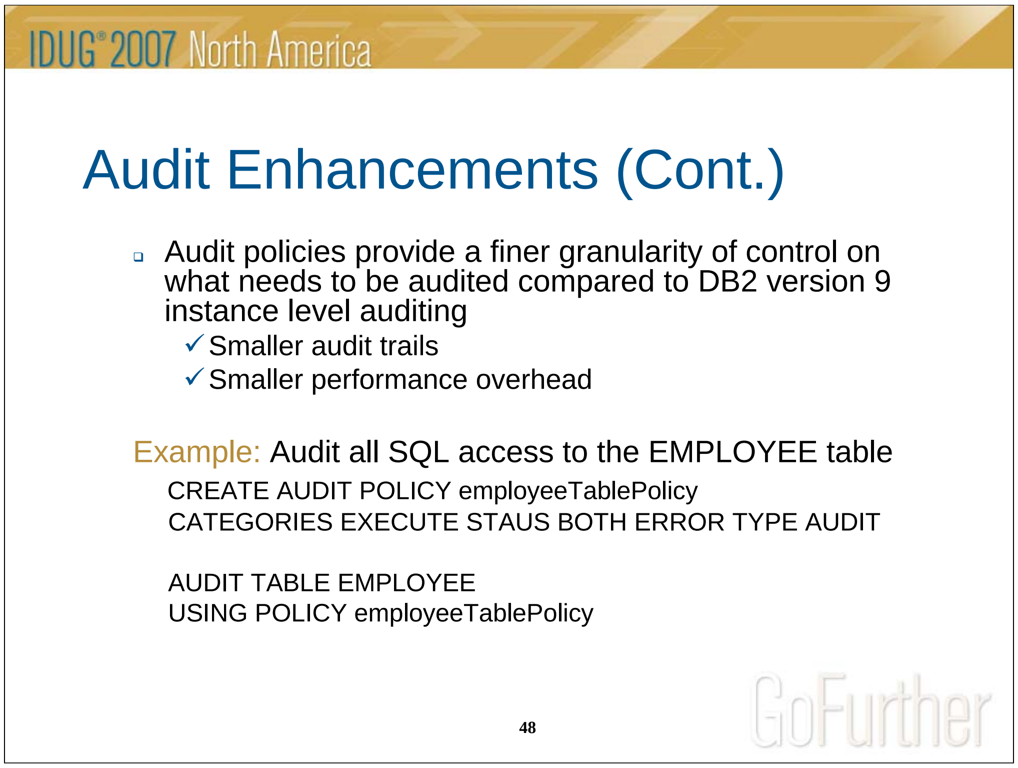
Audit Enhancements

- Audit Policy
  - A database object that specifies what categories of events are to be audited
  - An audit policy can be applied to:
    - ✓A database
    - ✓A table
    - ✓A trusted context
    - ✓An authorization id (user, role, group)
    - ✓An authority (SYSADM, SYSMAINT, SYSCTRL, SYSMON, DBADM, SECADM)

47

GoFurther

Audit policies is one the three key audit enhancements in VIPER II. It provides you with the granularity to audit only what you need. For example, you could choose to audit access by a specific user as opposed to auditing everyone else.


IDUG® 2007 North America

Audit Enhancements (Cont.)

- Audit policies provide a finer granularity of control on what needs to be audited compared to DB2 version 9 instance level auditing
  - ✓ Smaller audit trails
  - ✓ Smaller performance overhead

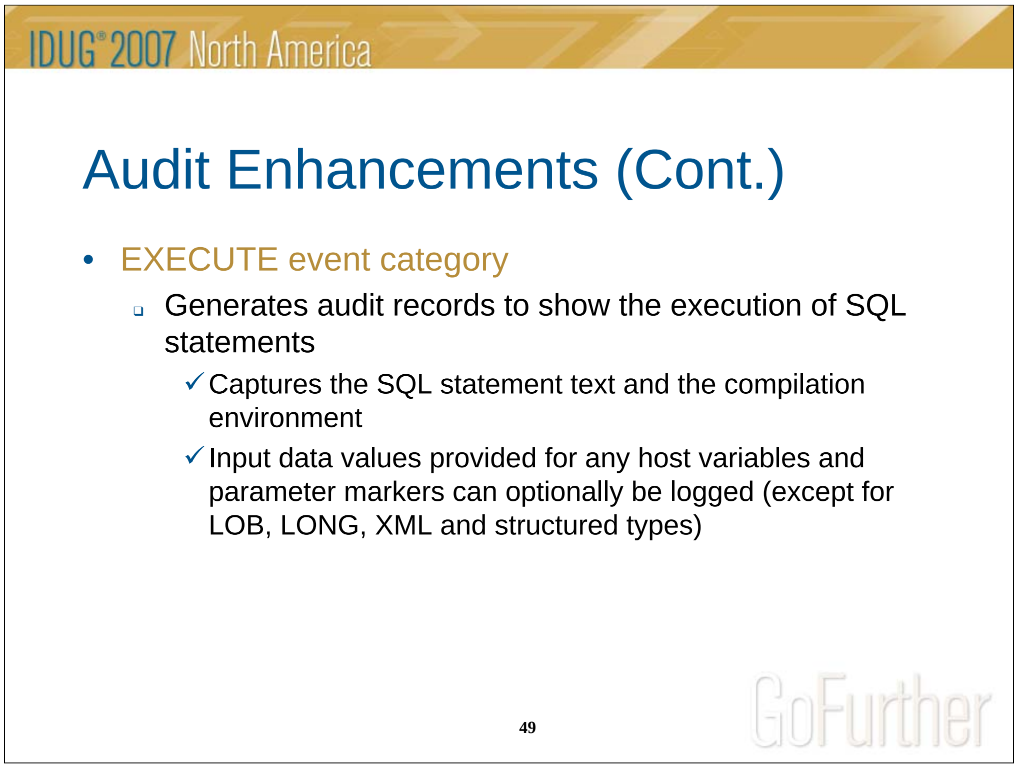
Example: Audit all SQL access to the EMPLOYEE table

```
CREATE AUDIT POLICY employeeTablePolicy
CATEGORIES EXECUTE STAUS BOTH ERROR TYPE AUDIT

AUDIT TABLE EMPLOYEE
USING POLICY employeeTablePolicy
```

48

GoFurther

The example shown illustrates the flexibility introduced through audit policies. In this example, the security administrator wishes to audit all SQL access to the EMPLOYEE table only.


IDUG® 2007 North America

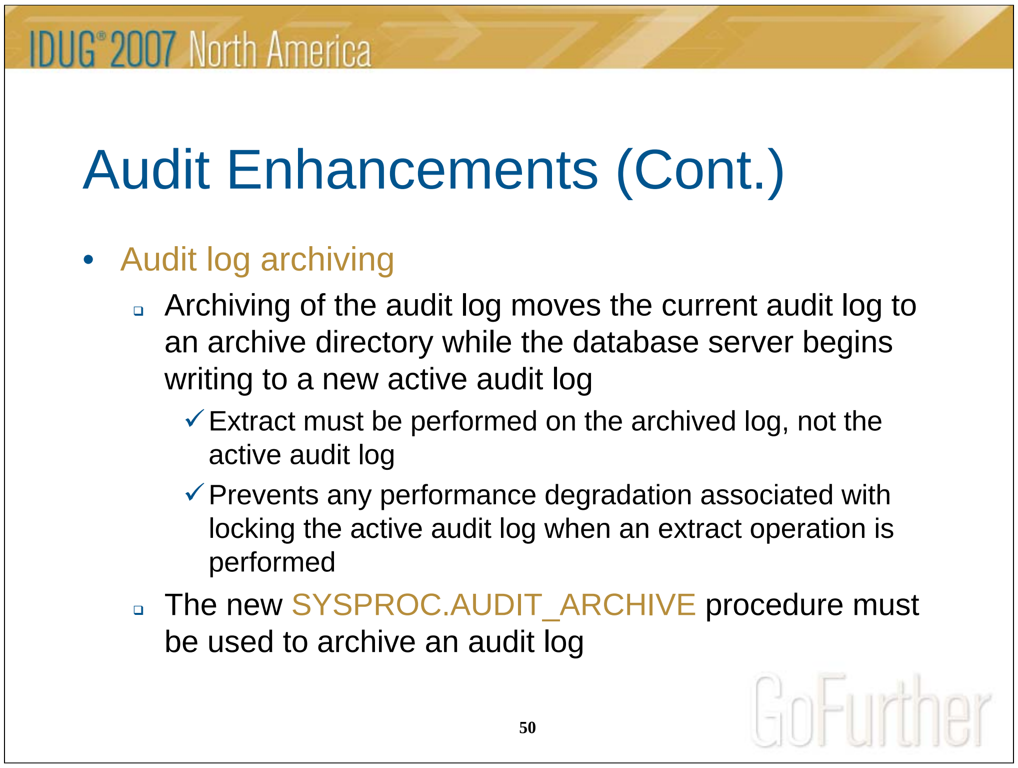
Audit Enhancements (Cont.)

- EXECUTE event category
  - Generates audit records to show the execution of SQL statements
    - ✓ Captures the SQL statement text and the compilation environment
    - ✓ Input data values provided for any host variables and parameter markers can optionally be logged (except for LOB, LONG, XML and structured types)

49

GoFurther

The EXECUTE category is another key audit enhancement in VIPER II. It allows to generate audit records to show the execution of SQL statements.



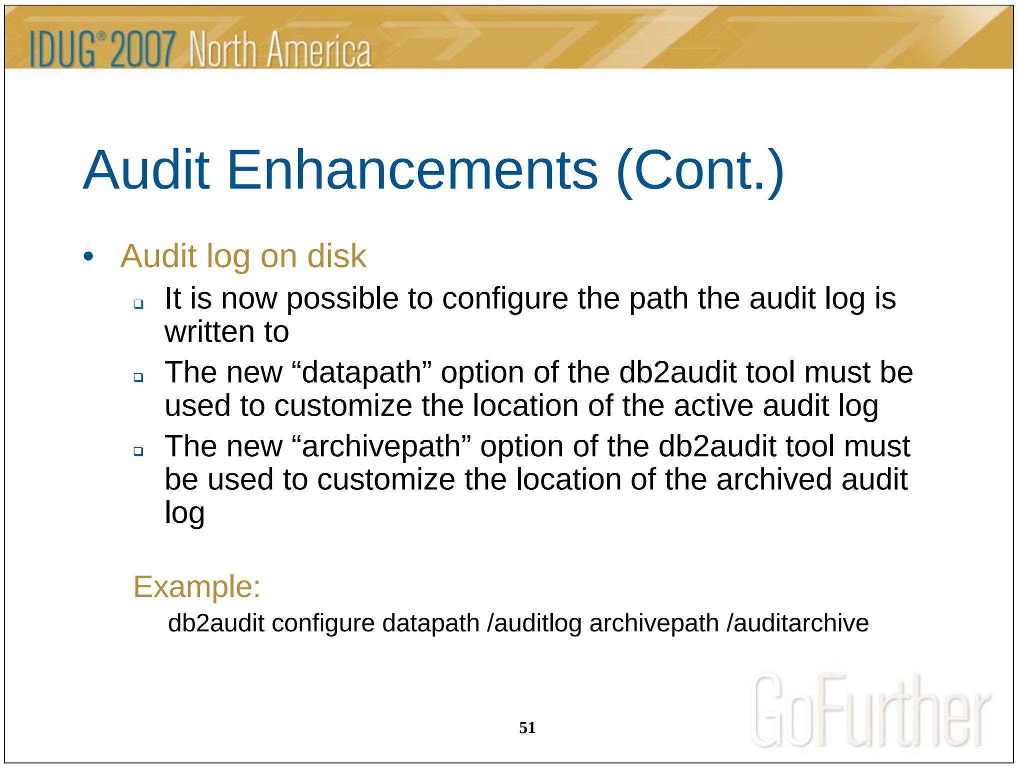
Audit Enhancements (Cont.)

- Audit log archiving
  - Archiving of the audit log moves the current audit log to an archive directory while the database server begins writing to a new active audit log
    - ✓ Extract must be performed on the archived log, not the active audit log
    - ✓ Prevents any performance degradation associated with locking the active audit log when an extract operation is performed
  - The new SYSPROC.AUDIT_ARCHIVE procedure must be used to archive an audit log

50

GoFurther

In VIPER II, it is possible to archive the audit log. In fact, the audit extract feature can only be performed on the archived log, not the active log.



Audit Enhancements (Cont.)

- Audit log on disk
  - It is now possible to configure the path the audit log is written to
  - The new “datapath” option of the db2audit tool must be used to customize the location of the active audit log
  - The new “archivepath” option of the db2audit tool must be used to customize the location of the archived audit log

Example:

```
db2audit configure datapath /auditlog archivepath /auditarchive
```

51

GoFurther

In previous releases of DB2, the audit log had to be in a predefined directory. In VIPER II, it is possible to customize the location where the audit log should reside.

IDUG® 2007 North America

Session: E13

DB2 for LUW Security: What's New in Version 9 and Beyond?

Walid Rjaibi

IBM

wrjaibi@ca.ibm.com

52

GoFurther