

# SurfDB Ein Token- basiertes Sicherheitskonzept - Grundlagen -

# Grundsätzliches

- ❖ Benutzerberechtigung kritischer Faktor
- ❖ Funktionalität schwer zu testen
- ❖ Logik der Berechtigung sollte weitestgehend von Applikationen unabhängig sein
- ❖ Datenaustausch zwischen Berechtigungslogik und Applikationen über ein standardisiertes Interface

# Lösungsansätze

- ❖ RowLevelSecurity mit db-Herstellermitteln
- ❖ Implementation innerhalb Applikationen
- ❖ LDAP / Active Directory

## Anforderung im Bereich DW

Heterogenes Umfeld unterstützen

Applikationen unterstützen

OLAP/direct access/data mining

## Lösung

surfDB

(secure user rights framework)

Sicherheit in DB DB-unabhängig

# Begriffe

- ❖ Applikationen
- ❖ Portal
- ❖ Berechtigungs-Dimension



Diese Dimensionen spannen  
den **Berechtigungsraum** auf

- Bildung von Datenbereichen
- Abhängigkeit der Dimensionen

# Definitionen und Beispiele

## Definition

**Token** sind atomare Rechte eines Benutzers, also Teilmengen eines Berechtigungsraumes.

## Beispiel

Der Berechtigungsraum habe 2 Dimensionen. Zum Beispiel Filiale und Abteilung.

Dann besagt der Token (1, 5) dass der Benutzer die Daten der Abteilung 5 in Filiale 1 sehen darf.

# Definitionen und Beispiele

## Definition

Beliebige Komponenten von Token dürfen den speziellen **Wert null** haben, der semantisch „alle“ bedeutet.

## Beispiel

Im vorherigen Beispiel besagt der Token (1, null), dass der Benutzer die Daten aller Abteilungen unterhalb der Filiale 1 sehen darf.

In Merkmal 2 (Abteilung) ist dann die Auswahl beliebig. Der Token (null, null) bedeutet, dass der Benutzer im Berechtigungsraum alle Rechte besitzt.

# Definitionen und Beispiele

## Definition

Die Anwendung fordert **Tokenlisten** vom Sicherheitsobjekt, die möglichst kurz sind.

## Beispiel

$[(1,5),(1,null)]$  ist keine gültige Tokenliste, da  $(1,null)$  die gleiche Bedeutung hat und kürzer ist.

Wenn der Token  $(null,null)$  existiert, so ist er stets das einzige Element in der Tokenliste.

# Drei Arten von Token

- ❖ Funktionale Token: definieren die Applikationen, auf die ein Benutzer Zugriff haben soll. Sie haben eine simple Struktur, die aus jeweils einem Identifikator besteht.
- ❖ Datentoken: definieren den weitaus komplexeren Bereich der Datensicherheit in den verschiedenen Datenbereichen.
- ❖ Usertoken: definieren eventuelle weitere Rechte oder Einschränkungen des Benutzers, die quer zur hierarchischen Datenstruktur gelten.

# SurfDB - Ein Token- basiertes Sicherheitskonzept - Technische Umsetzung -

# Voraussetzungen

- ❖ Imaginäre Firma mit 3 Filialen
- ❖ Jede Filiale besteht aus 2 Abteilungen
- ❖ Ziel: Umsatzanalyse
- ❖ Benutzer können Rechte auf einzelne Abteilungen, ganze Filialen oder alle Daten haben

# Nutzdaten

```
CREATE TABLE umsatz (  
    filiale SMALLINT,  
    abteilung SMALLINT,  
    monat SMALLINT,  
    umsatz DECIMAL(10,2)  
);
```

# Token aller User

```
CREATE TABLE Token (  
  UserID CHAR(20) NOT NULL,  
  filiale INTEGER,  
  abteilung INTEGER  
);
```

# Minimalität der Tokenliste

```
DELETE FROM Token T WHERE EXISTS
  (SELECT * FROM Token UT WHERE
    UT.UserID = T.UserID
    AND (UT.filiale = T.filiale OR
    UT.filiale IS NULL)
    AND (UT.abteilung = T.abteilung OR
    UT.abteilung IS NULL)
    AND (UT.RowID != T.RowID) )
```

# Angemeldete Benutzer

```
CREATE TABLE CURRENT_USER (  
    SessionId INTEGER;  
    UserID CHAR (20) );
```

## Benutzer anmelden

```
CREATE PROCEDURE set_current_user(uid CHAR(20))  
    DEFINE sid INTEGER;  
    LET sid=DBINFO("sessionid");  
    DELETE FROM current_user WHERE SessionID=sid;  
    INSERT INTO current_user VALUES (sid, uid);  
END PROCEDURE
```

# Extraktion der Token eines Users

```
CREATE VIEW User_Token AS  
  SELECT * FROM Token T, CURRENT_USER C  
  WHERE T.UserID = C.UserID  
  AND C.SessionID=DBINFO(`sessionid`)  
UNION  
  SELECT * FROM Token T  
  WHERE T.UserID = USER;
```

# Zugriff auf Benutzerdaten



```
CREATE VIEW U_Umsatz AS
  SELECT Umsatz.* FROM Umsatz U, USER_TOKEN T
  WHERE (U.filiale=T.filiale
         OR T.filiale IS NULL )
  AND (U.abteilung=T.abteilung
       OR T.abteilung IS NULL );
```

Beziehungsweise, wenn kein direkter Join möglich ist:

```
CREATE VIEW USER_N AS
  SELECT * FROM N
  WHERE EXISTS ( SELECT * FROM USER_T
                WHERE ( N.nt1=USER_T.t1 OR USER_T.t1 IS NULL)
                AND (N.t2 = USER_T.t2 OR USER_T.t2 IS NULL) );
```



Dirk Gunsthövel GunCon - IT Systemanalyse

Hammer Straße 13, 48153 Münster

Tel.: +49-(0)251- 28 44 6-0, Fax: +49-(0)251- 28 44 6-55, Mobil: +49-(0)170- 5464715

[Dirk@GunCon.de](mailto:Dirk@GunCon.de)



