

ISO/IEC JTC 1/SC 32 N 0426

Date: 2000-01-25

REPLACES: --

<p style="text-align: center;">ISO/IEC JTC 1/SC 32</p> <p style="text-align: center;">Data Management and Interchange</p> <p style="text-align: center;">Secretariat: United States of America (ANSI) Administered by Pacific Northwest National Laboratory on behalf of ANSI</p>

DOCUMENT TYPE	Other Document (Open)
TITLE	Working Group 3 Tutorial - Santa Fe
SOURCE	Krishna Kulkarni and Jim Melton
PROJECT NUMBER	
STATUS	This will assist in the understanding of the WG 3 work
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	
DUE DATE	
Number of Pages	45
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile: +1 703 681 9180; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite <http://bwonotes5.wdc.pnl.gov/SC32/JTC1SC32.nsf>

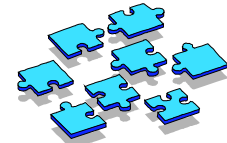
*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI



Working Group 3 Tutorial

Krishna Kulkarni and
Jim Melton, USA

Presented to SC32 Working Group
Tutorial Meeting on 2000-01-24



▼ Working Group 3

- Title: Database Languages
- Area of Work (Ref: SC32 N249)
 - Develop and maintain languages for the dynamic specification, maintenance and description of database structures and contents in multi-user and multi-server environments. May include
 - Data types, behaviors and integrity constraints on the contents of the defined structures.
 - Mechanisms for the creation and generation of new data types and behaviors
 - Develop and maintain languages that provide for the storage, access and manipulation of data in database structures by multiple concurrent users.
 - Provide interfaces for the languages developed to other standard programming languages
 - Provide interfaces or access to other standards describing data types, behaviors or database content to users of the languages developed.
- Convenor: Stephen Cannan
- Editor: Jim Melton

WG3 Published Standards

- Database Language SQL
 - ▶ First Edition published in 1987 as ISO 9075-1987
 - ▶ Second Edition published in 1989 as ISO/IEC 9075:1989
 - ▶ Third Edition published in 1992 as ISO/IEC 9075:1992
 - ▶ Fourth Edition published in 1999 as
 - **ISO/IEC 9075-1:1999 (SQL/Framework)**
 - **ISO/IEC 9075-2:1999 (SQL/Foundation)**
 - **ISO/IEC 9075-3:1999 (SQL/CLI)**
 - **ISO/IEC 9075-4:1999 (SQL/PSM)**
 - **ISO/IEC 9075-5:1999 (SQL/Bindings)**

WG3 Project List

- FCD Editing Meeting in progress
 - SQL/OLB (Object Language Binding)
- Currently under FCD ballot
 - SQL/MED (Management of External Data)
- Current under FPDAM ballot
 - Amendment 1 - SQL/OLAP
- Working Drafts
 - SQL/Framework
 - SQL/Foundation
 - SQL/CLI
 - SQL/PSM
 - SQL/Temporal
 - SQL/Schemata



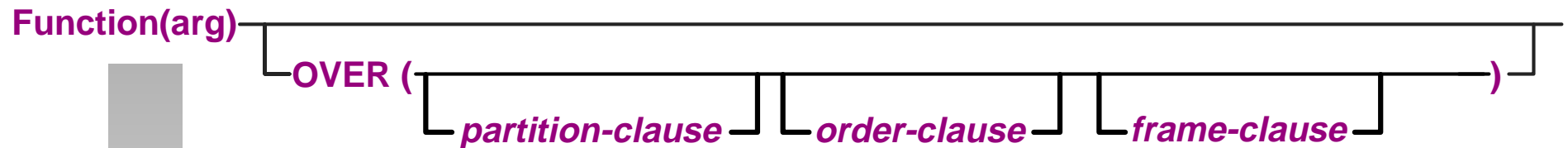
SQL/OLAP

- First formal *amendment* to SQL-99
 - ▶ Expected to be published by late 2000
 - ▶ Specifies 34 new built-in functions:
 - 7 new numeric functions
 - 16 new aggregate functions
 - 5 new windowed table functions
 - 4 new hypothetical aggregate functions
 - 2 new inverse distribution functions
 - ▶ Also specifies windowed-table versions of both old and new aggregate functions
 - ▶ Significant functionality and performance advantages for OLAP applications


▼ Windowed Table functions

- A **windowed table function** operates on a window of a table and returns a value for every row in that window.
- 5 new windowed table functions
 - ▶ RANK () OVER ...
 - ▶ DENSE_RANK () OVER ...
 - ▶ PERCENT_RANK () OVER ...
 - ▶ CUME_DIST () OVER ...
 - ▶ ROW_NUMBER () OVER ...

▼ Windowed Table Functions



- Windows are defined using the ***window-clause***.
- Window clause consists of
 - A partition clause - The partition clause allows the rows in a table to be grouped into partitions.
 - An order clause - The order clause allows the rows in a partition to be ordered.
 - A frame clause - The frame clause allows the specification of the range of rows relative to the current row that would participate in the function evaluation.



Rank, Dense_Rank, and Rownumber

■ RANK

- ▶ Returns the relative position within an ordered list.
- ▶ Requires ordering.
- ▶ Ties have the same rank.

■ DENSE_RANK

- ▶ Like RANK, but no gaps in rankings in the case of ties.

■ ROW_NUMBER

- ▶ Ties are nondeterministically numbered.
- ▶ If no ordering is specified, each row is nondeterministically numbered.

Rank, Dense_Rank, and Rownumber

- ▶ Find the ranking of each employee in descending order of salary.

```
select empnum, dept, salary,  
       rank() over (order by salary desc) as rank,  
       dense_rank() over (order by salary desc) as denserank,  
       row_number() over (order by salary desc) as rownum  
from emptab;
```

EMPNUM	DEPT	SALARY	RANK	DENSERANK	ROWNUM
3	-	84000	1	1	1
8	3	79000	2	2	2
6	1	78000	3	3	3
2	1	75000	4	4	4
7	1	75000	4	4	5
12	3	75000	4	4	6
10	3	55000	7	5	7
11	1	53000	8	6	8

...

Ranking null values

- Find the ranking of each employee in descending order of salary.

```
select empnum, dept, salary,  
       rank() over (order by salary desc nulls last) as ranknl  
from emptab;
```

EMPNUM	DEPT	SALARY	RANKNL
3	-	84000	1
8	3	79000	2
6	1	78000	3
2	1	75000	4
7	1	75000	4
12	3	75000	4
10	3	55000	7
11	1	53000	8
5	1	52000	9
9	2	51000	10
1	1	50000	11
0	-	-	12
4	2	-	12

Ranking within a partition

- ▶ Find the ranking of each employee in descending order of salary within the employee's department.

```
select empnum, dept, salary,  
       rank() over (partition by dept order by salary desc nulls last)  
       as rank_in_dept,  
       rank() over (order by salary desc nulls last) as globalrank  
from emptab;
```

EMPNUM	DEPT	SALARY	RANK_IN_DEPT	GLOBALRANK
6	1	78000	1	3
2	1	75000	2	4
7	1	75000	2	4
11	1	53000	4	8
5	1	52000	5	9
1	1	50000	6	11
9	2	51000	1	10
4	2	-	2	12
8	3	79000	1	2
12	3	75000	2	4
10	3	55000	3	7
3	-	84000	1	1
0	-	-	2	12

▼ Aggregate Functions as Windowed Table Functions

- Existing aggregate functions can also be used as windowed table functions:
 - ▶ SUM (...) OVER ...
 - ▶ AVG (...) OVER ...
 - ▶ MAX (...) OVER ...
 - ▶ MIN (...) OVER ...
 - ▶ COUNT (...) OVER ...
 - ▶ EVERY (...) OVER ...
 - ▶ ANY (...) OVER ...
 - ▶ SOME (...) OVER ...
- Allows calculation of moving and cumulative aggregate values.

Cumulative Aggregates

- ▶ Find the cumulative sum of salaries of employees ordered by their employee number.

```
select empnum, dept, salary,  
       sum(salary) over (order by empnum asc) as cume_sal  
from emptab;
```

EMPNUM	DEPT	SALARY	CUME_SAL
1	2	84000	84000
2	3	79000	163000
3	1	78000	241000
4	1	75000	316000
5	1	75000	391000
6	3	75000	466000
7	3	55000	521000
8	2	53000	574000

▼ Moving Aggregates

- ▶ Find the moving average of salaries of employees ordered by their employee number averaged over the given employee and the employee with immediately higher employee number.

```
select empnum, dept, salary,  
       avg(salary) over (order by empnum asc rows 1 preceding) as  
       mov_sal  
from emptab;
```

EMPNUM	DEPT	SALARY	MOV_SAL
1	2	84000	84000
2	3	79000	81500
3	1	78000	78500
4	1	75000	76500
5	1	75000	75000
6	3	75000	75000
7	3	55000	65000
8	2	53000	54000



New Numeric Functions

- ▶ LN (expr)
- ▶ EXP (expr)
- ▶ POWER (expr, expr)
- ▶ SQRT (expr)
- ▶ FLOOR (expr)
- ▶ CEIL[ING] (expr)
- ▶ WIDTH_BUCKET(expr, expr, expr, expr) EX: WIDTH_BUCKET (age,0,100,10)



New Aggregate Functions

- ▶ REGR_SLOPE (expr, expr)
- ▶ REGR_INTERCEPT (expr, expr)
- ▶ REGR_COUNT (expr, expr)
- ▶ REGR_R2 (expr, expr)
- ▶ REGR_AVGX (expr, expr)
- ▶ REGR_AVGY (expr, expr)
- ▶ REGR_SXX (expr, expr)
- ▶ REGR_SYY (expr, expr)
- ▶ REGR_SXY (expr, expr)

▼ New Aggregate Functions (contd.)

- ▶ STDDEV_POP (expr)
- ▶ STDDEV_SAMP (expr)
- ▶ VAR_POP (expr)
- ▶ VAR_SAMP (expr)
- ▶ COVAR_POP (expr, expr)
- ▶ COVAR_SAMP (expr, expr)
- ▶ CORR (expr, expr)

▼ Hypothetical Aggregate Functions

- Hypothetical aggregate functions evaluate the aggregate over the window extended with a new row derived from the specified values.
- 4 new hypothetical aggregate functions
 - ▶ RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
 - ▶ DENSE_RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
 - ▶ PERCENT_RANK (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)
 - ▶ CUME_DIST (expr, expr ...) WITHIN GROUP (ORDER BY <sort specification list>)

▼ Hypothetical Aggregate Functions

- ▶ Find the RANK, DENSE_RANK, PERCENT_RANK and CUME_DIST of a house whose price is \$400,000 among houses ordered on price.

```
SELECT Area,  
RANK (400000) WITHIN GROUP (ORDER BY Price DESC),  
DENSE_RANK (400000) WITHIN GROUP (ORDER BY Price DESC),  
PERCENT_RANK (400000) WITHIN GROUP (ORDER BY Price DESC),  
CUME_DIST (400000) WITHIN GROUP (ORDER BY Price DESC)  
FROM Homes;
```

Inverse Distribution Functions

- 2 new inverse distribution functions
 - PERCENTILE_DISC (expr) WITHIN GROUP (ORDER BY <sort specification list>)
 - PERCENTILE_CONT (expr) WITHIN GROUP (ORDER BY <sort specification list>)
- Argument must evaluate to a value between 0 and 1.
- Return the values of expressions specified in <sort specification list> that correspond to the specified percentile value.

▼ Inverse Distribution Functions

- ▶ Find the 50th percentile price for the homes.

```
SELECT Area,  
  PERCENTILE_CONT (0.5)  
    WITHIN GROUP (ORDER BY Price DESC),  
  PERCENTILE_DISC (0.5)  
    WITHIN GROUP (ORDER BY Price DESC)  
FROM Homes;
```



Working Group 3 Tutorial (Continued)

SQL/MED

Management of External Data



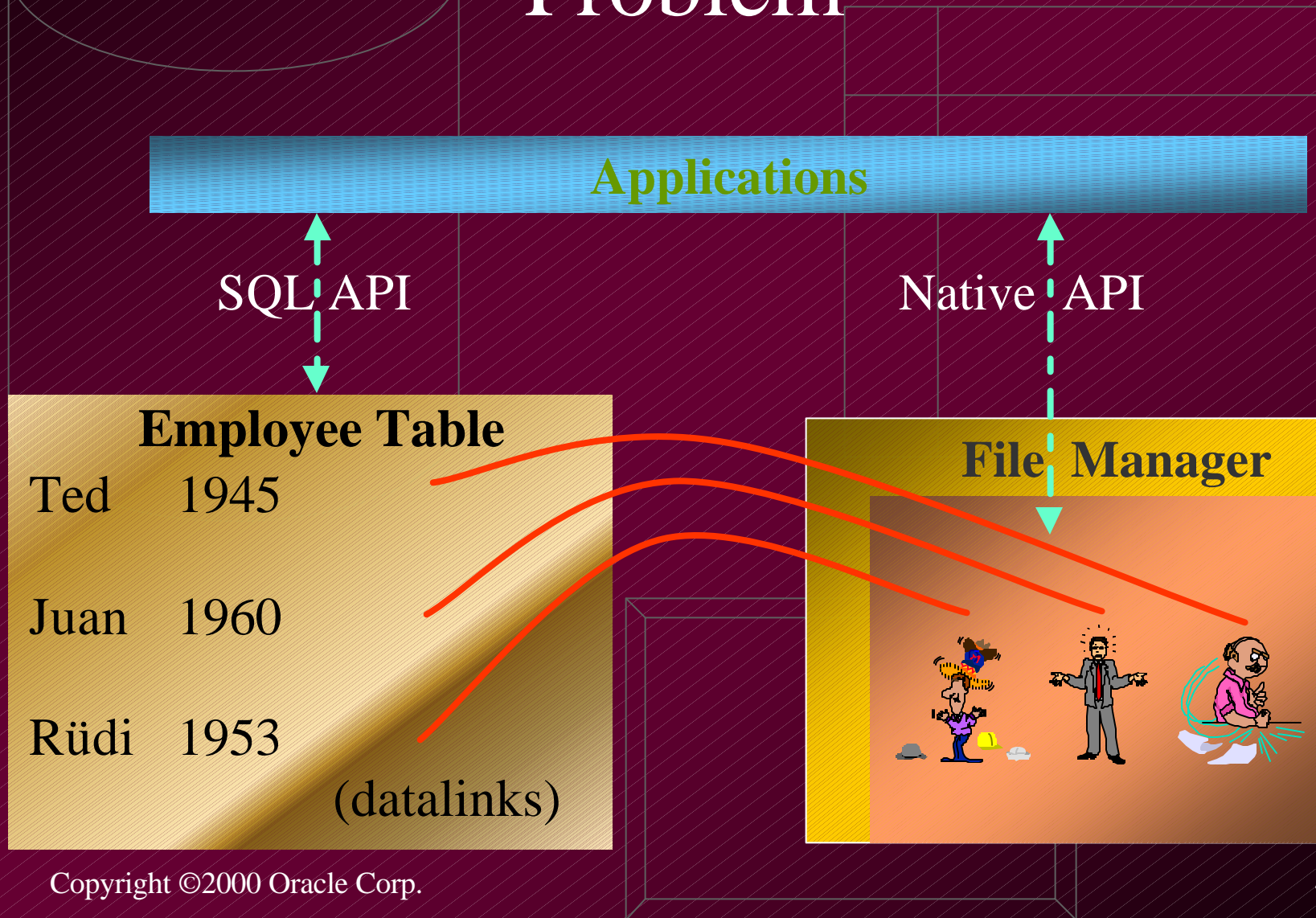
Two Kinds of External Data

- Coordinated by SQL-server (**Datalink**)
 - Kept consistent with SQL-data
 - Accessed through *native* interface
- Accessed through SQL-server (**Foreign Tables**)
 - Avoid native interface
 - Mix-and-match data sources, data *unified* by SQL-server

Datalinks: The Management Problem

- “Links” data files with SQL-data
- Referential integrity, recovery, and backup coordinated with database data
- Authorization to access data handled by the database manager *or* data owner
- Native application interface used to open and access the data

Datalinks: The Management Problem



Defining DATALINK Columns

```
CREATE TABLE movies (  
  Title          CHARACTER VARYING(50),  
  RunningTime   INTEGER,  
  Movie         DATALINK  
    FILE LINK CONTROL  
    READ PERMISSION DB  
    INTEGRITY ALL  
    WRITE PERMISSION BLOCKED  
    ON UNLINK RESTORE )
```

Inserting DATALINK Values

- Links the file to the database
- Invoke DLVALUE constructor

```
INSERT INTO Movies
```

```
( Title, RunningTime, Movie )
```

```
VALUES ( 'My Life', 126, DLVALUE(  
    'http://www.movies.com/mylife.avi',  
    'URL',  
    'The story of a standards rep' ) )
```

Deleting DATALINKs

- Examples assume FILE LINK CONTROL is specified for the column
- DELETE unlinks the file

```
DELETE
```

```
FROM Movies
```

```
WHERE Title = 'My Life'
```

Updating DATALINKs

- UPDATE unlinks existing file and replaces with link to new file

```
UPDATE Movies
```

```
  SET Movie =
```

```
    DLVALUE (
```

```
      'http://server1.foo.org/me.mpg',
```

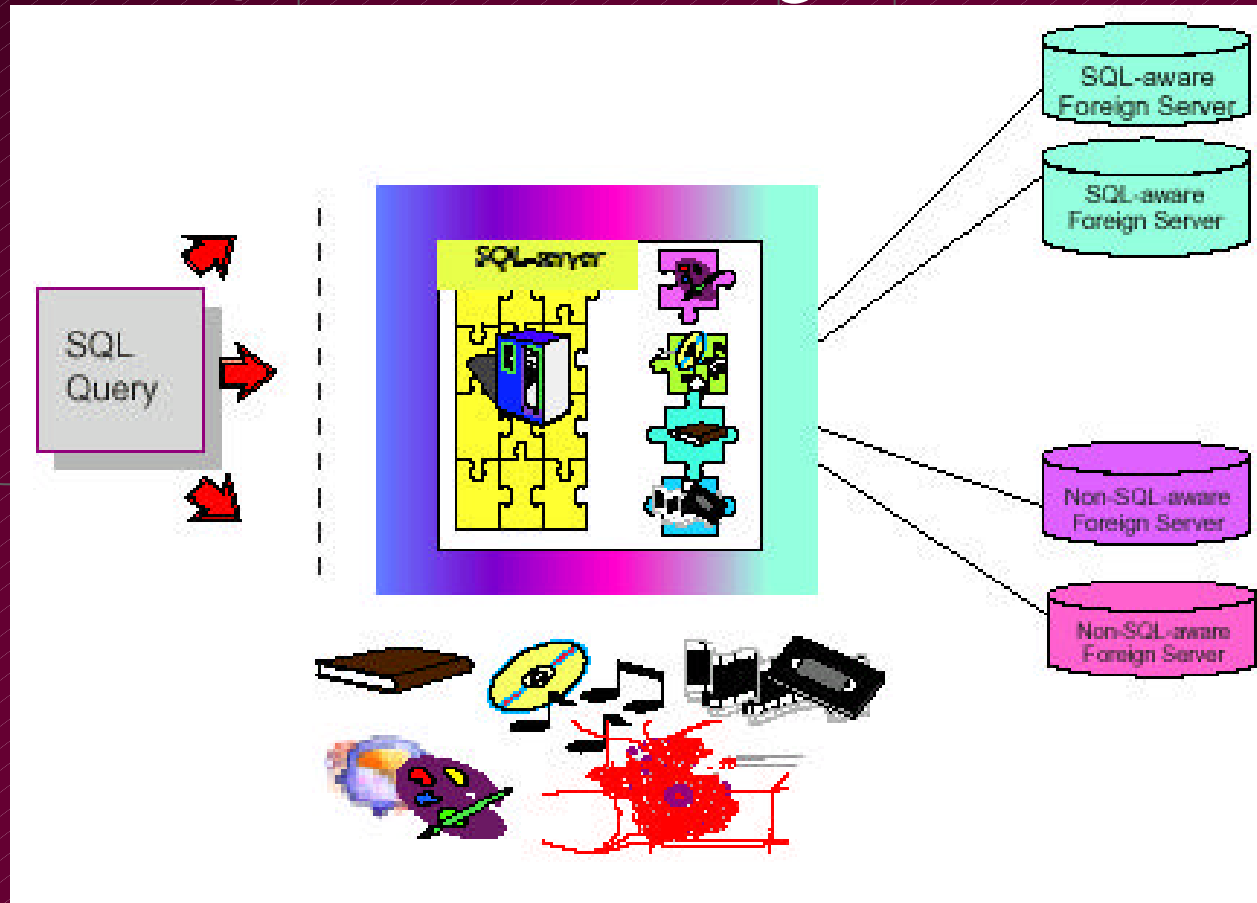
```
      'URL', 'A standards rep''s life')
```

```
WHERE Title='My Life'
```

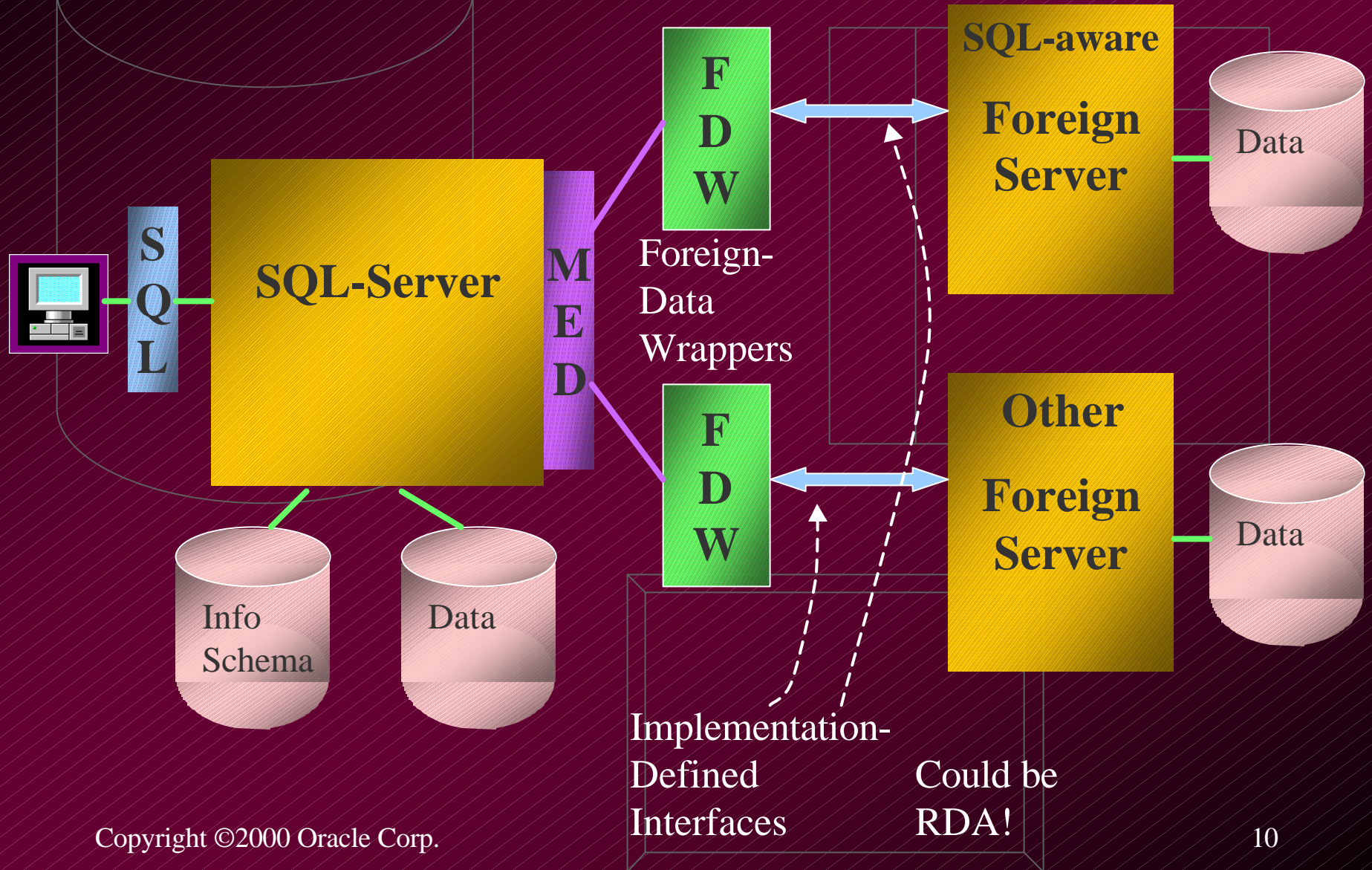
- Old file ownership/permissions restored

Foreign Tables

- Purpose: SQL API to heterogeneous data



SQL/MED Architecture



Wrapper Interface Terminology

- Foreign Server—Entity outside the SQL environment that manages its own data
- Foreign-Data Wrapper—A collection of routines used to access a foreign server
- User Mapping—Maps an SQL authorization identifier to a corresponding concept at a foreign server

More Terminology

- Generic Options—A set of “option name, option value” pairs.
 - Foreign servers
 - Foreign-data wrappers
 - User mappings
 - Foreign tables and their columns
 - Not (yet) standardized, but some might be needed
 - Generally the responsibility of the “foreign” objects and *not* of the (local) SQL-server

New SQL Statements

- **CREATE FOREIGN DATA WRAPPER**
 wrapper_name
 LIBRARY "..." LANGUAGE **x**
 OPTIONS (...)
- **CREATE SERVER server_name**
 FOREIGN DATA WRAPPER wrapper_name
 OPTIONS (URL "...", ...)
- **CREATE FOREIGN TABLE table_name (**
 column-definition, ...)
 SERVER server_name
 OPTIONS (...)

Sample Use of SQL/MED

- Tell me the customers and stores that are located in an area for which the weather forecast predicts humidity of 90% or greater and temperatures of 95°F or greater:

```
SELECT s.store_id, c.name, c.address
FROM customers AS c, stores AS s,
     weather AS w
WHERE humidity_forecast(w.zone)>=90 AND
     temperature_forecast(w.zone)>=85 AND
     within(c.address, w.zone) AND
     within(s.address, w.zone)
```

Foreign-Data Wrapper Interface

- Allocate and deallocate resources
- Control connections to foreign servers
- Receive information from SQL-server about the SQL statement that it wants the wrapper to execute at the foreign servers
- Send information to SQL-server about aspects of that statement the wrapper can execute
- Initiate and terminate execution of statements

SQL-Server/Wrapper Dialog

SQL-Server	Foreign-Data Wrapper
<ul style="list-style-type: none">• Wrapper, initialize yourself; here's some information you need to know	<ul style="list-style-type: none">• What information do I need to know about myself?
<ul style="list-style-type: none">• Here's some information you need to know about the current user and the foreign server. Go ahead and connect to that foreign server.	<ul style="list-style-type: none">• OK, let me see what you're telling me about the foreign server and the user. I'm connecting now and here is the handle you must use when working through this connection.
<ul style="list-style-type: none">• Here's what I need...	<ul style="list-style-type: none">• That's interesting...here's what I can do in that respect.
<ul style="list-style-type: none">• OK, I'll make my plans around your capabilities.	
<ul style="list-style-type: none">• Get ready to execute...now go!	<ul style="list-style-type: none">• I'm sending you data until it's all done.
<ul style="list-style-type: none">• Thanks...free your resources	<ul style="list-style-type: none">• OK, finished.

Handles

- Integer values that identify encapsulated information
- Some allocated by SQL-server, some by foreign-data wrapper
- “Get” and “set” routines to manage contents of described data structures/areas

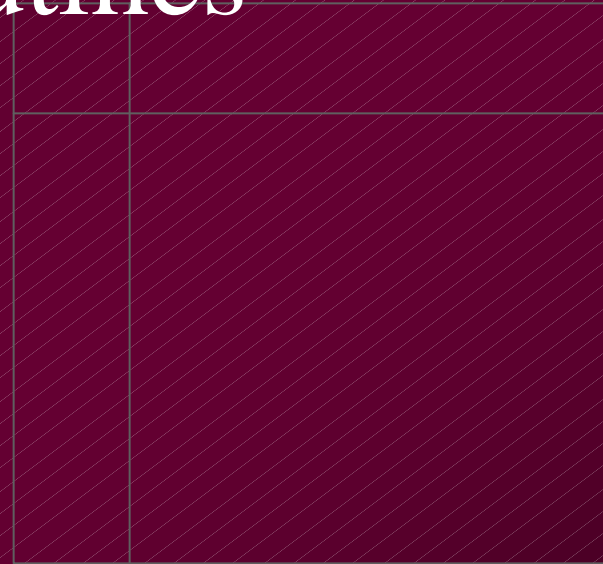
Types of Handles

- WrapperEnv
- Wrapper
- Server
- FSConnection
- Request

- Table reference
- Value expression
- Reply
- Execution
- User

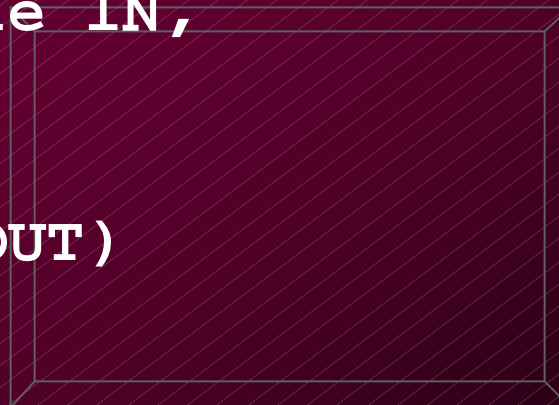
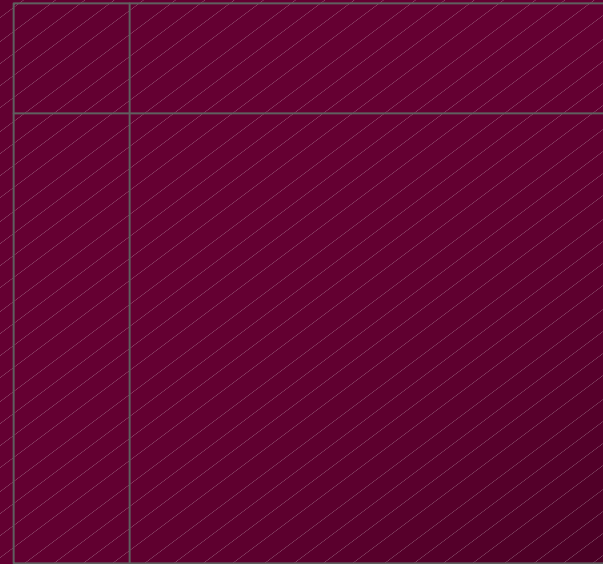
Foreign-Data Wrapper Interface Routines

- Initialization routines
- Access routines
- Termination routines
- Handle routines



Initialization Routines

- `AllocWrapperEnv`
(`WrapperHandle` IN,
`WrapperEnvHandle` OUT)
- `ConnectForeignServer`
(`WrapperEnvHandle` IN,
`ServerHandle` IN,
`FSConnectionHandle` OUT)
- `InitForeignRequest`
(`FSConnectionHandle` IN,
`RequestHandle` IN,
`ReplyHandle` OUT,
`ExecutionHandle` OUT)



Access Routines

- **Open (ExecutionHandle IN)**
- **Iterate (ExecutionHandle IN)**
- **ReOpen (ExecutionHandle IN)**
- **Close (ExecutionHandle IN)**
- **Note that these routines effectively emulate operations on cursors**

Termination Routines

- **FreeReplyHandle**
(ReplyHandle IN)
- **FreeExecutionHandle**
(ExecutionHandle IN)
- **FreeFSConnection**
(FSConnectionHandle IN)
- **FreeWrapperHandle**
(WrapperHandle IN)

Handle Routines

- Server Handle Routines
- Wrapper Handle Routines
- User Handle Routines
- Request Handle Routines
- Table Reference Handle Routines
- Value Expression Handle Routines
- Reply Handle Routines

