Using extensibility in everyday applications

Kevin Brown, Informix Lead Architect, IBM,
kbrown3@us.ibm.com

C14
Wednesday April 30th 10:50-11:50 AM

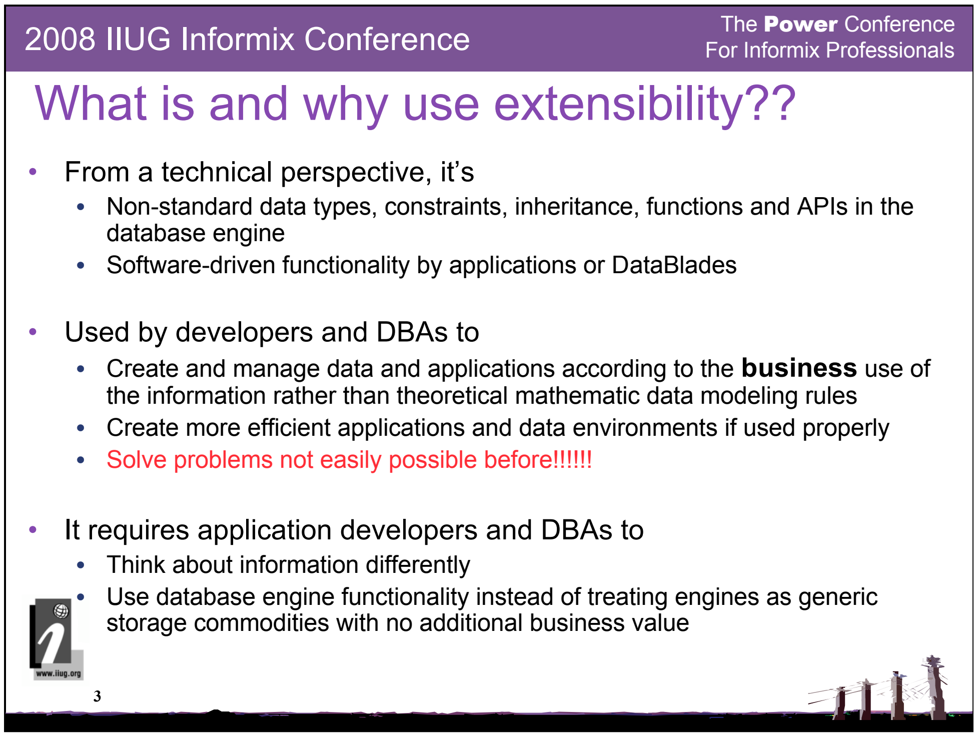
2008 IIUG Informix Conference

www.iiug.org

Version Date: Oct 18, 2001 8:58 pm SKC(CD)

Agenda

- What is and why use extensibility?
- Using “extended” data types in applications
- A quick look at DataBlades



2008 IIUG Informix Conference

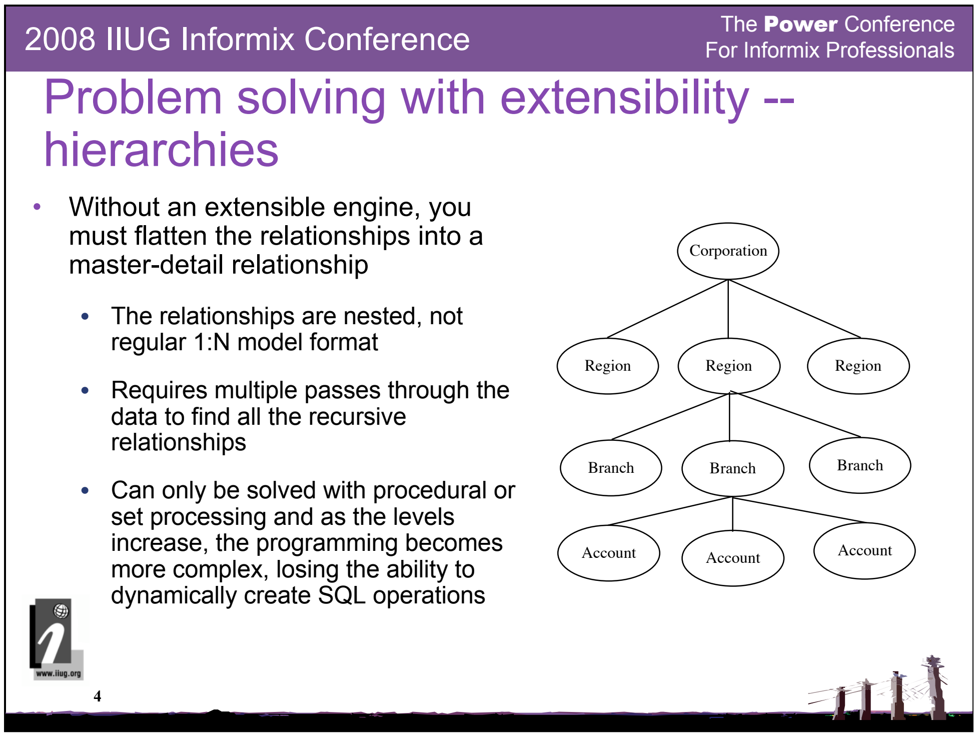
The **Power** Conference For Informix Professionals

What is and why use extensibility??

- From a technical perspective, it's
  - Non-standard data types, constraints, inheritance, functions and APIs in the database engine
  - Software-driven functionality by applications or DataBlades
- Used by developers and DBAs to
  - Create and manage data and applications according to the **business** use of the information rather than theoretical mathematic data modeling rules
  - Create more efficient applications and data environments if used properly
  - Solve problems not easily possible before!!!!!!
- It requires application developers and DBAs to
  - Think about information differently
  - Use database engine functionality instead of treating engines as generic storage commodities with no additional business value

www.iiug.org

3

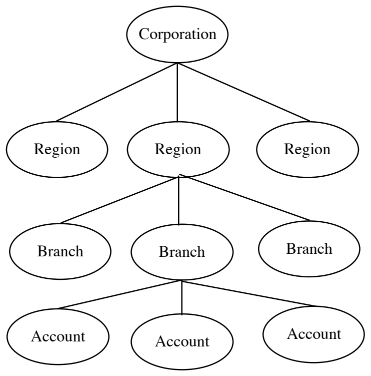


2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

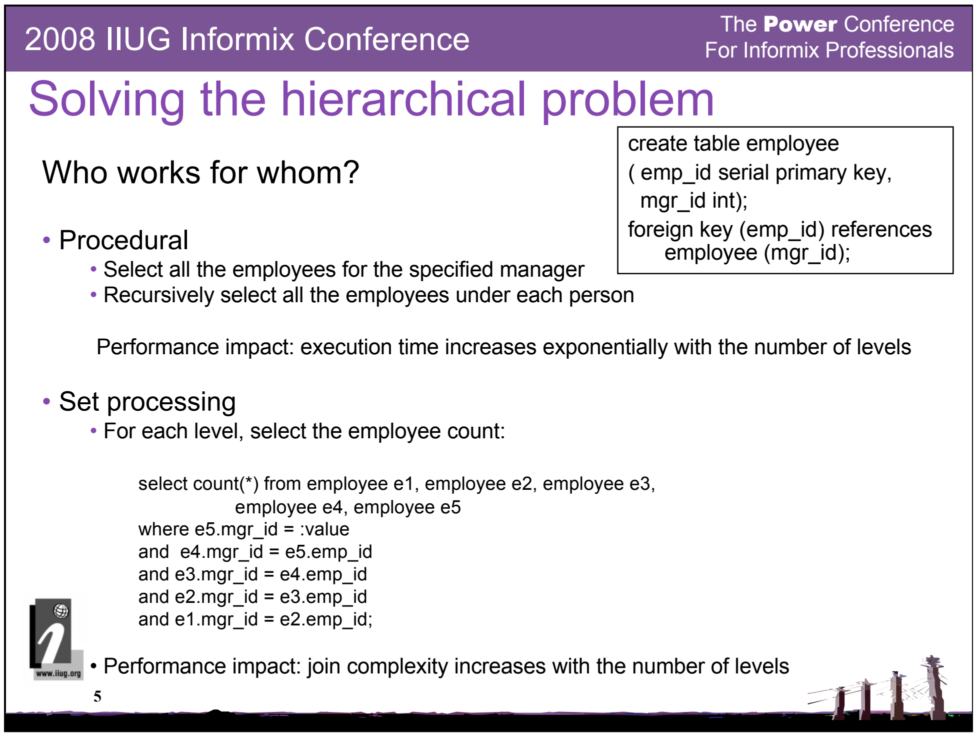
Problem solving with extensibility -- hierarchies

- Without an extensible engine, you must flatten the relationships into a master-detail relationship
  - The relationships are nested, not regular 1:N model format
  - Requires multiple passes through the data to find all the recursive relationships
  - Can only be solved with procedural or set processing and as the levels increase, the programming becomes more complex, losing the ability to dynamically create SQL operations



www.iiug.org

4



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Solving the hierarchical problem

```
create table employee
( emp_id serial primary key,
  mgr_id int);
foreign key (emp_id) references
      employee (mgr_id);
```

Who works for whom?

- Procedural
  - Select all the employees for the specified manager
  - Recursively select all the employees under each person

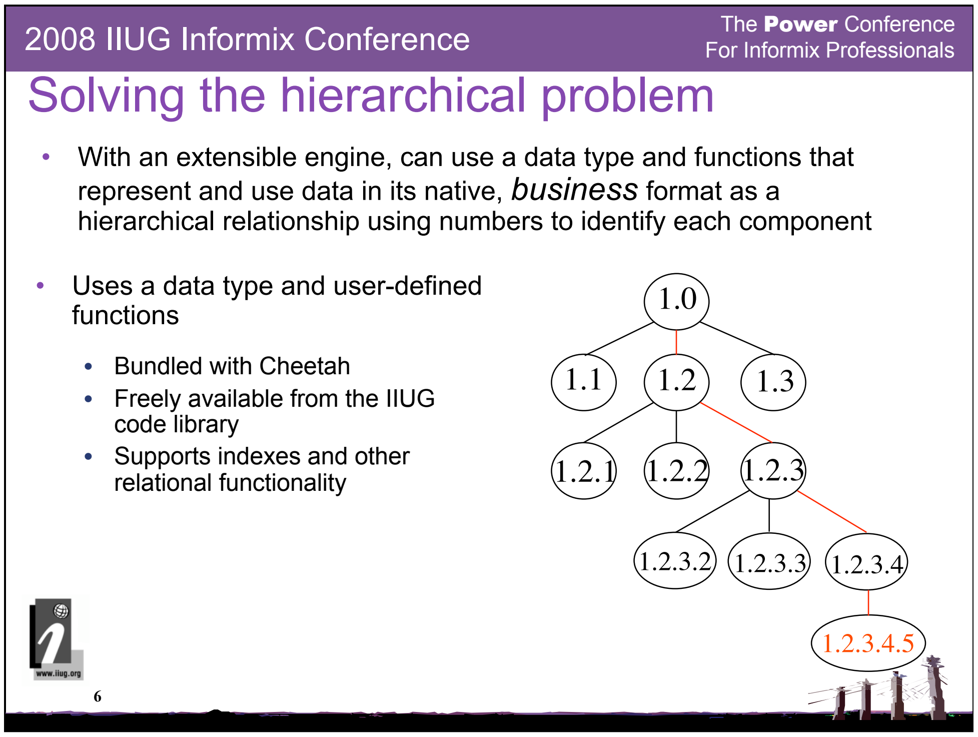
  Performance impact: execution time increases exponentially with the number of levels

- Set processing
  - For each level, select the employee count:

```
select count(*) from employee e1, employee e2, employee e3,
                employee e4, employee e5
where e5.mgr_id = :value
and  e4.mgr_id = e5.emp_id
and e3.mgr_id = e4.emp_id
and e2.mgr_id = e3.emp_id
and e1.mgr_id = e2.emp_id;
```

  - Performance impact: join complexity increases with the number of levels

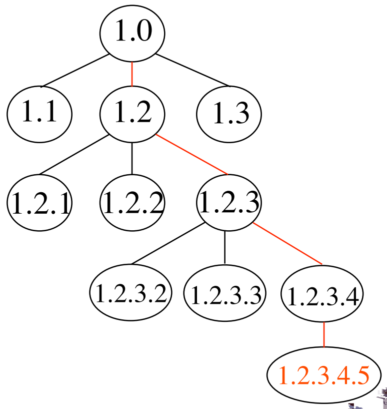
5

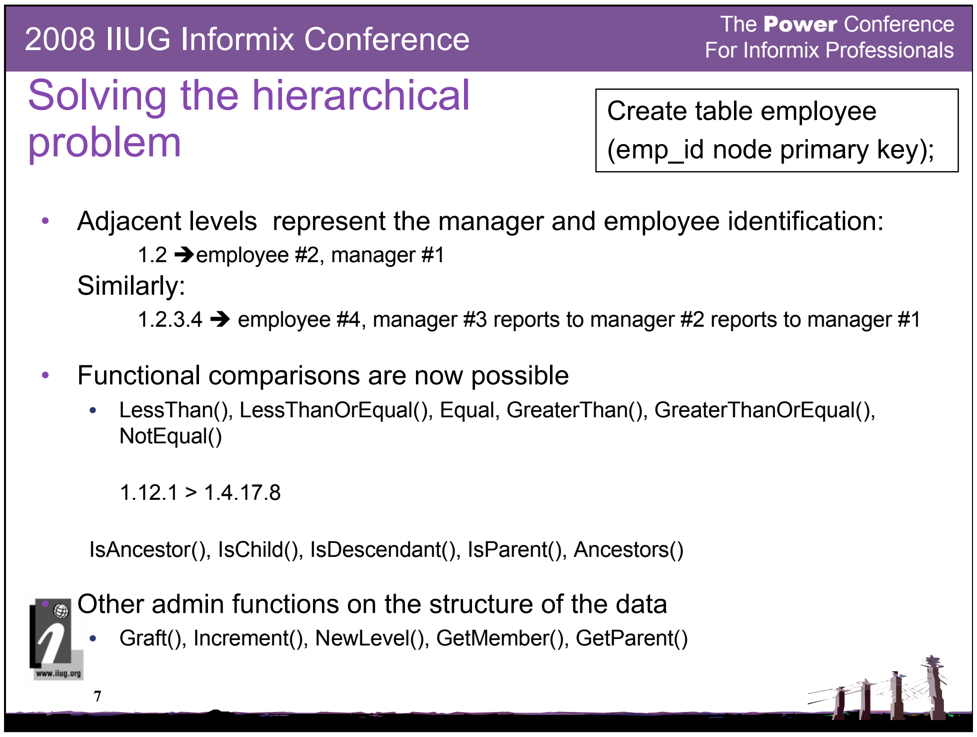


Solving the hierarchical problem

- With an extensible engine, can use a data type and functions that represent and use data in its native, *business* format as a hierarchical relationship using numbers to identify each component

- Uses a data type and user-defined functions
  - Bundled with Cheetah
  - Freely available from the IIUG code library
  - Supports indexes and other relational functionality





Solving the hierarchical problem

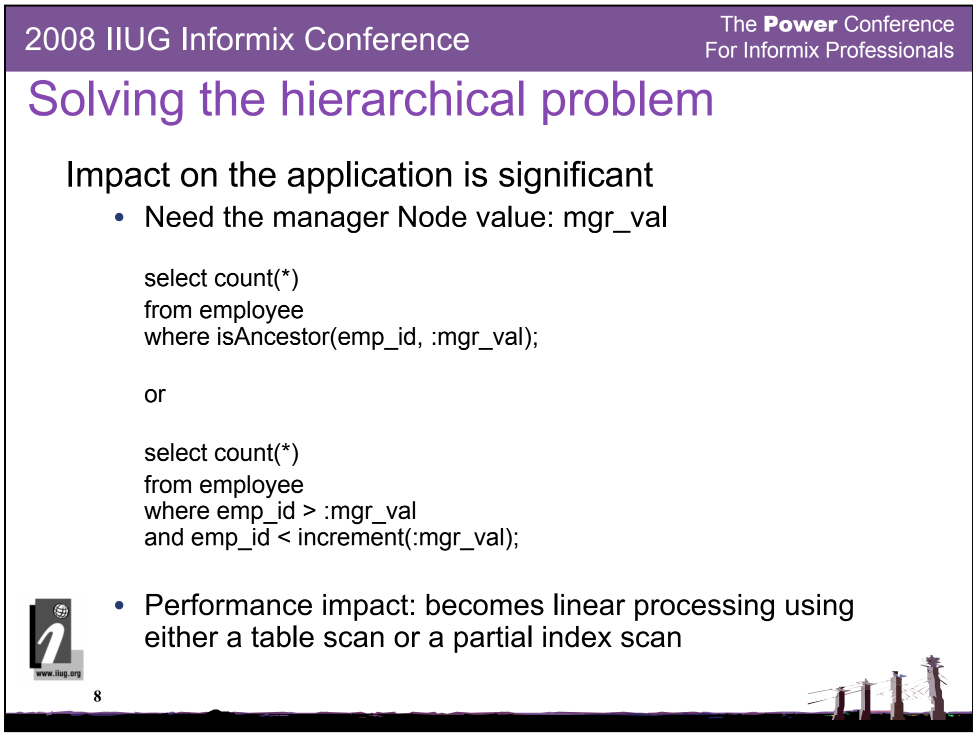
Create table employee (emp_id node primary key);

- Adjacent levels represent the manager and employee identification:
  - 1.2 ➔employee #2, manager #1

  Similarly:
  - 1.2.3.4 ➔ employee #4, manager #3 reports to manager #2 reports to manager #1
- Functional comparisons are now possible
  - LessThan(), LessThanOrEqual(), Equal, GreaterThan(), GreaterThanOrEqual(), NotEqual()

    1.12.1 > 1.4.17.8

  IsAncestor(), IsChild(), IsDescendant(), IsParent(), Ancestors()
- Other admin functions on the structure of the data
  - Graft(), Increment(), NewLevel(), GetMember(), GetParent()



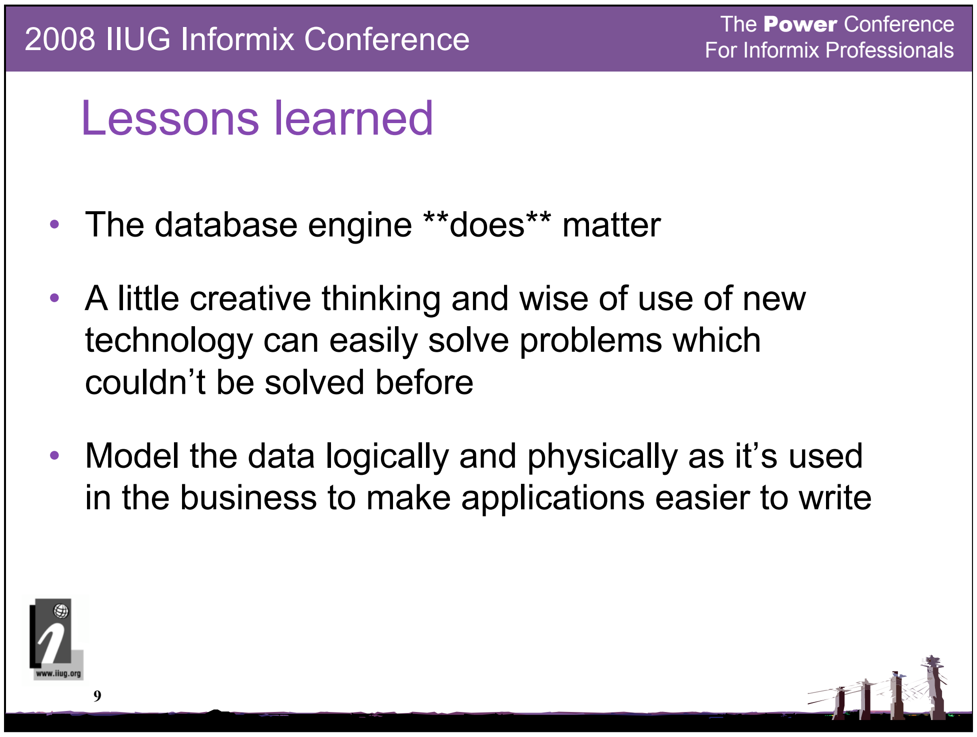
Solving the hierarchical problem

Impact on the application is significant

- Need the manager Node value: mgr_val

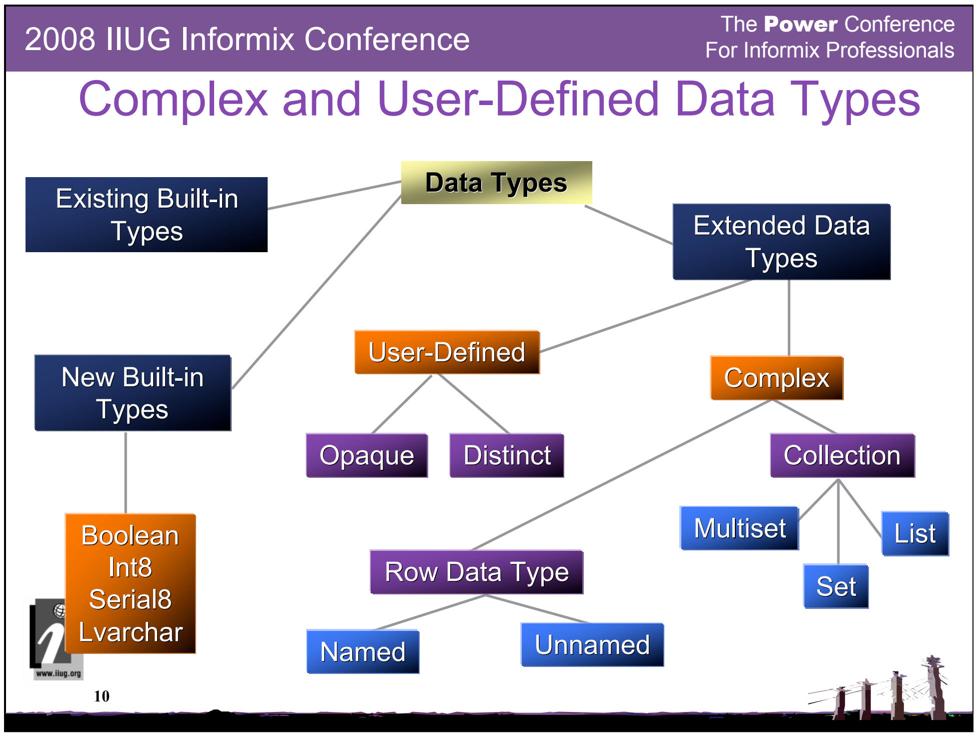
```
select count(*)
from employee
where isAncestor(emp_id, :mgr_val);

or

select count(*)
from employee
where emp_id > :mgr_val
and emp_id < increment(:mgr_val);
```

- Performance impact: becomes linear processing using either a table scan or a partial index scan



Lessons learned

- The database engine **does** matter
- A little creative thinking and wise of use of new technology can easily solve problems which couldn’t be solved before
- Model the data logically and physically as it’s used in the business to make applications easier to write

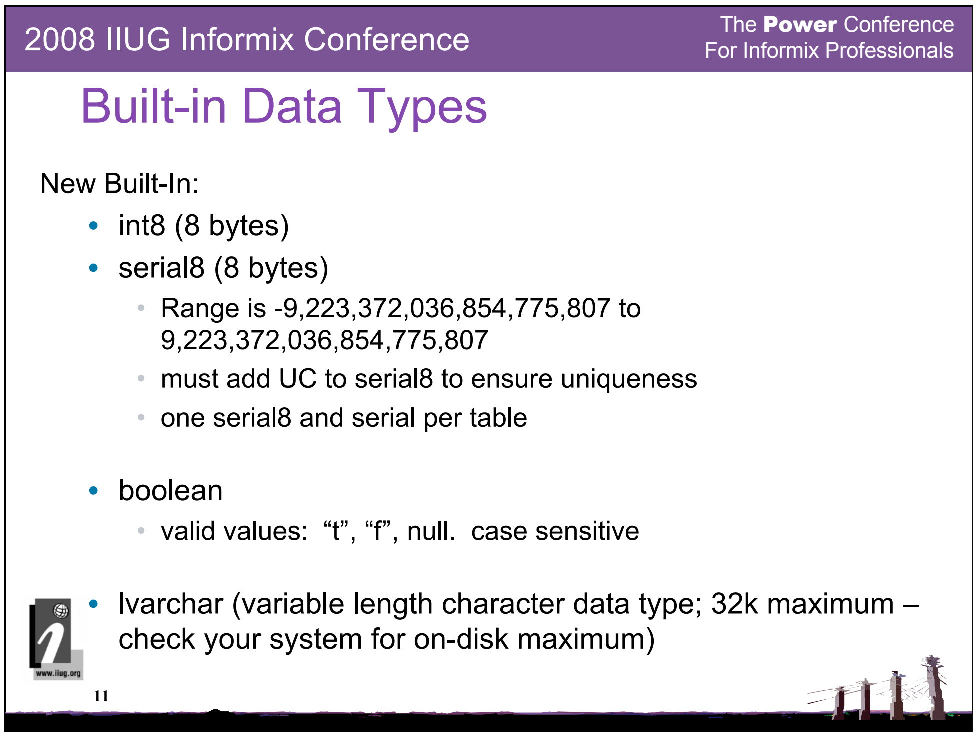


2008 IIUG Informix Conference

The **Power** Conference For Informix Professionals

Complex and User-Defined Data Types

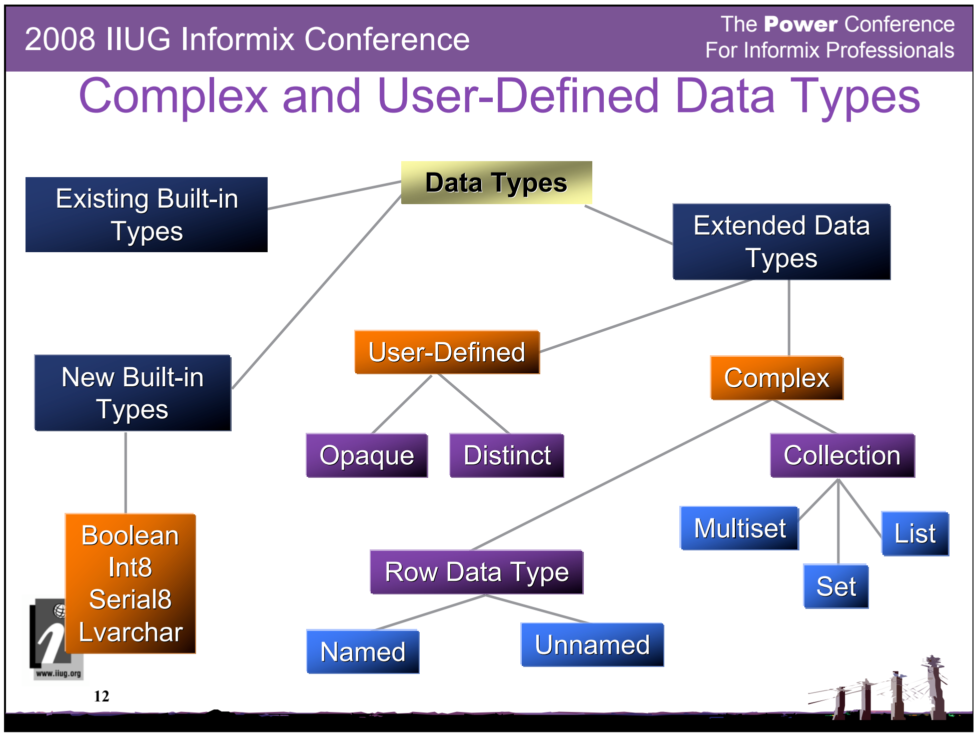
10



Built-in Data Types

New Built-In:

- int8 (8 bytes)
- serial8 (8 bytes)
  - Range is -9,223,372,036,854,775,807 to 9,223,372,036,854,775,807
  - must add UC to serial8 to ensure uniqueness
  - one serial8 and serial per table
- boolean
  - valid values: “t”, “f”, null. case sensitive
- lvarchar (variable length character data type; 32k maximum – check your system for on-disk maximum)

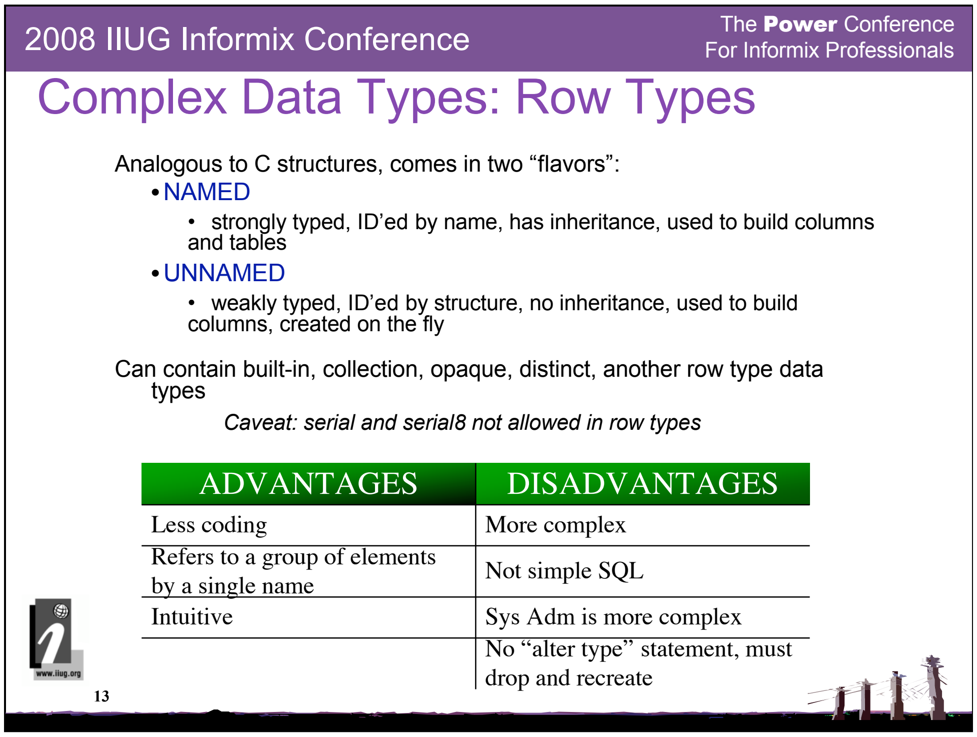


2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Complex and User-Defined Data Types

12



Complex Data Types: Row Types

Analogous to C structures, comes in two “flavors”:

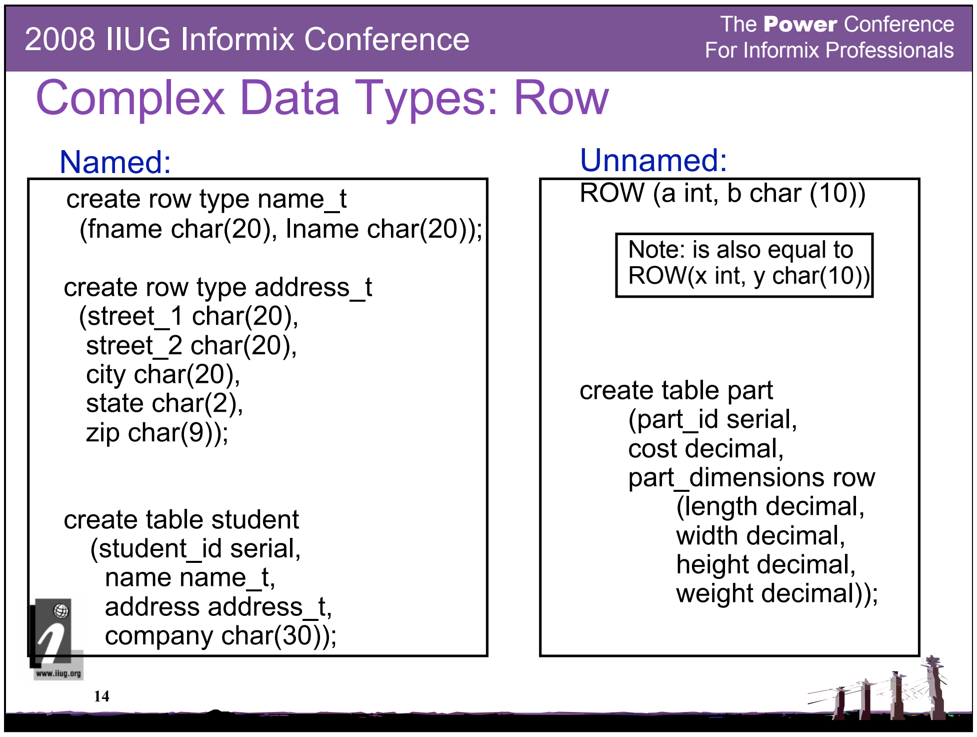
- NAMED
  - strongly typed, ID’ed by name, has inheritance, used to build columns and tables
- UNNAMED
  - weakly typed, ID’ed by structure, no inheritance, used to build columns, created on the fly

Can contain built-in, collection, opaque, distinct, another row type data types

Caveat: serial and serial8 not allowed in row types

ADVANTAGES	DISADVANTAGES
Less coding	More complex
Refers to a group of elements by a single name	Not simple SQL
Intuitive	Sys Adm is more complex
	No “alter type” statement, must drop and recreate

www.iiug.org



Complex Data Types: Row

Named:

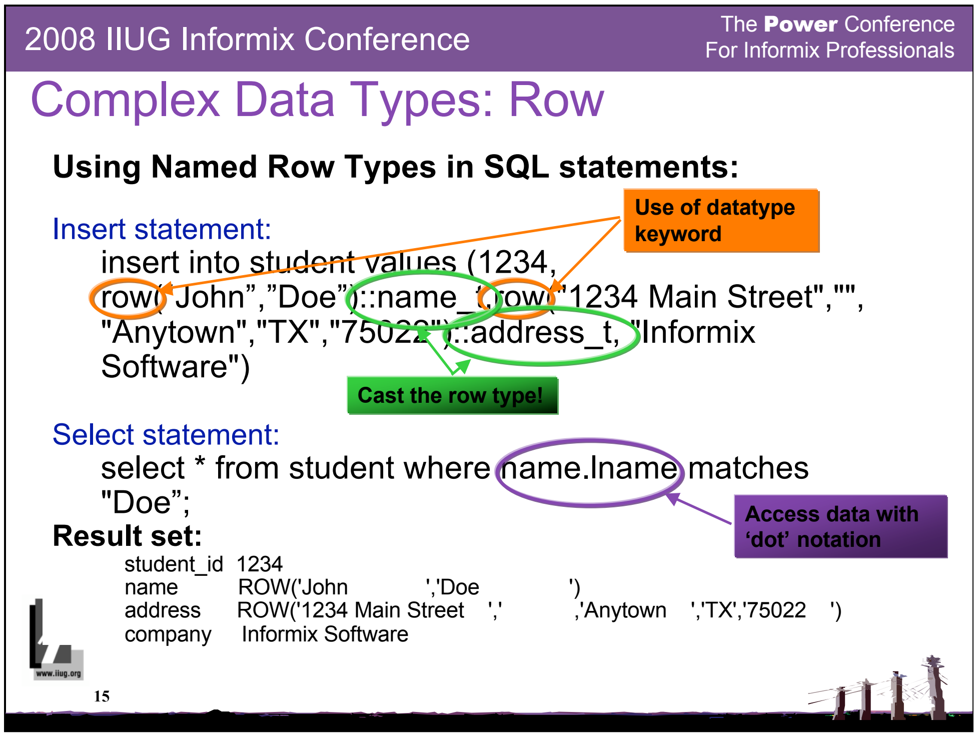
```
create row type name_t
  (fname char(20), lname char(20));

create row type address_t
  (street_1 char(20),
   street_2 char(20),
   city char(20),
   state char(2),
   zip char(9));


create table student
    (student_id serial,
      name name_t,
      address address_t,
      company char(30));
```

Unnamed:

```
ROW (a int, b char (10))
```

Note: is also equal to
ROW(x int, y char(10))

```
create table part
      (part_id serial,
      cost decimal,
      part_dimensions row
            (length decimal,
            width decimal,
            height decimal,
            weight decimal));
```



Complex Data Types: Row

Using Named Row Types in SQL statements:

Insert statement:

```
insert into student values (1234,
row("John","Doe")::name_t, row("1234 Main Street","",
"Anytown","TX","75022")::address_t, "Informix
Software")
```

Use of datatype keyword

Cast the row type!

Select statement:

```
select * from student where name.lname matches
"Doe";
```

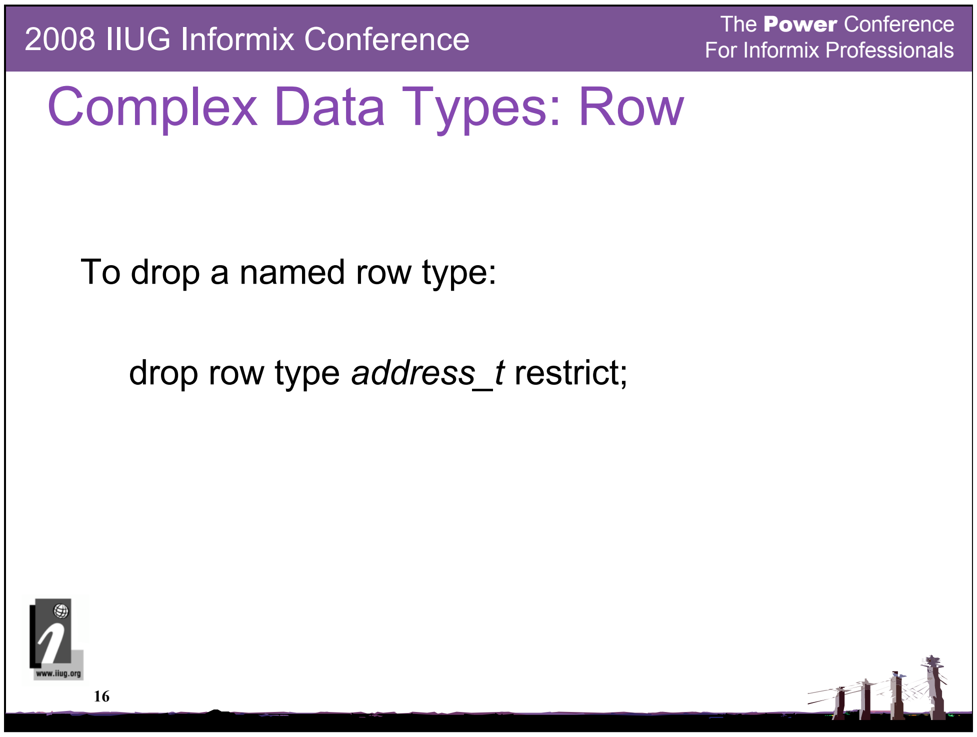
Access data with 'dot' notation

Result set:

```
student_id  1234
name        ROW('John          ','Doe            ')
address     ROW('1234 Main Street   ','          ,'Anytown    ','TX','75022    ')
company      Informix Software
```

www.iiug.org

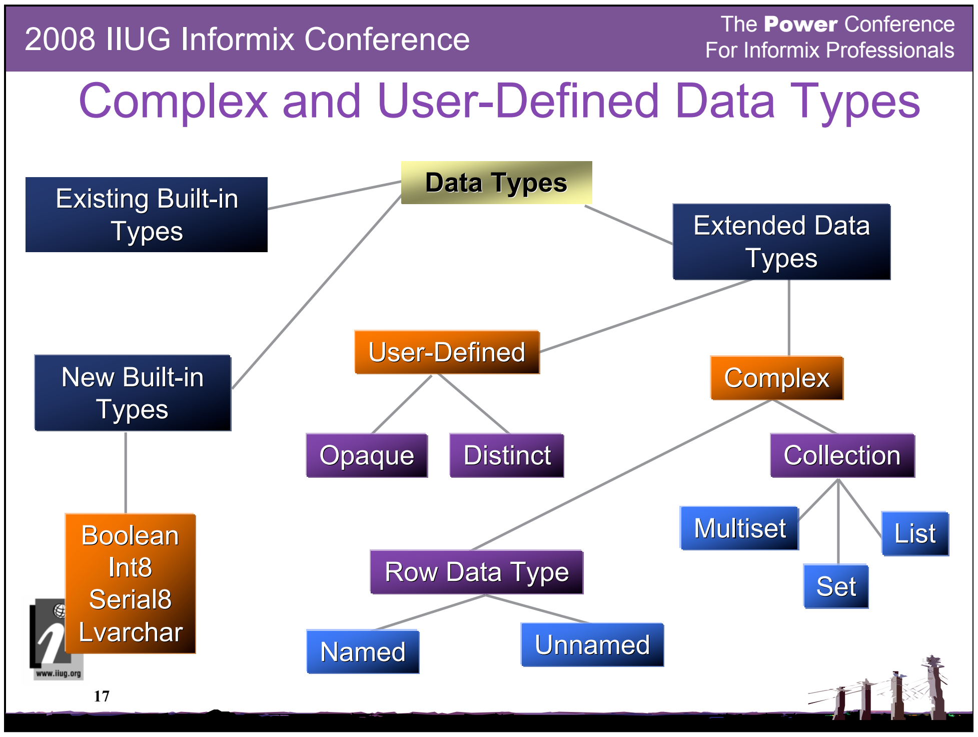
15



Complex Data Types: Row

To drop a named row type:

drop row type *address_t* restrict;

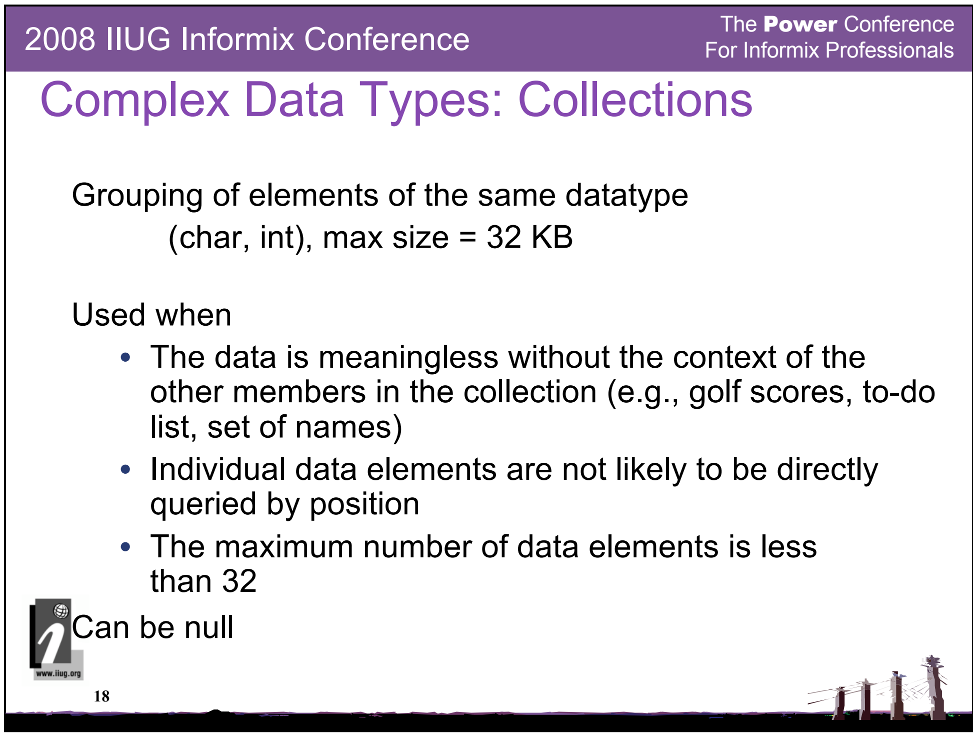


2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Complex and User-Defined Data Types

17



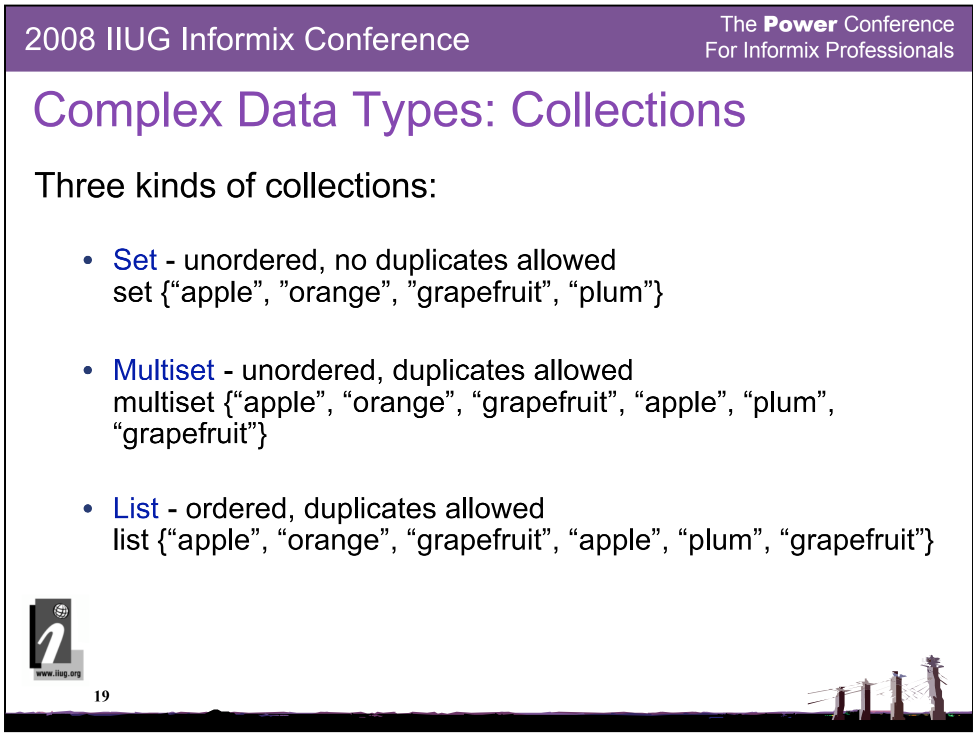
Complex Data Types: Collections

Grouping of elements of the same datatype
(char, int), max size = 32 KB

Used when

- The data is meaningless without the context of the other members in the collection (e.g., golf scores, to-do list, set of names)
- Individual data elements are not likely to be directly queried by position
- The maximum number of data elements is less than 32

Can be null



Complex Data Types: Collections

Three kinds of collections:

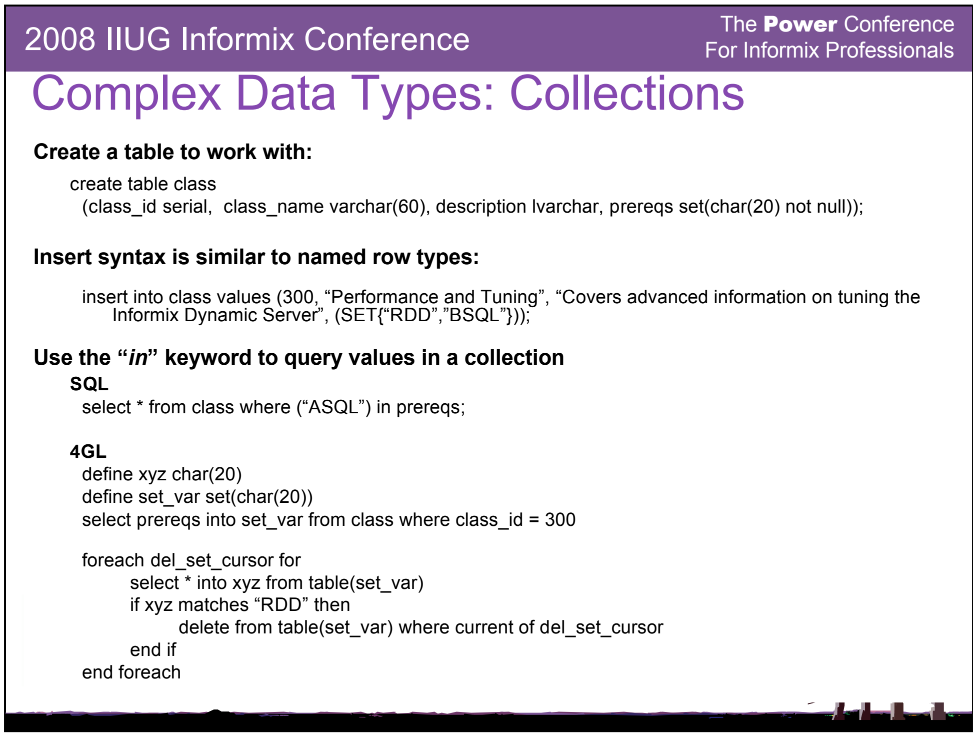
- Set - unordered, no duplicates allowed
  set {“apple”, ”orange”, ”grapefruit”, “plum”}

- Multiset - unordered, duplicates allowed
  multiset {“apple”, “orange”, “grapefruit”, “apple”, “plum”, “grapefruit”}

- List - ordered, duplicates allowed
  list {“apple”, “orange”, “grapefruit”, “apple”, “plum”, “grapefruit”}

www.iiug.org

19



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Complex Data Types: Collections

Create a table to work with:

```
create table class
  (class_id serial,  class_name varchar(60), description lvarchar, prereqs set(char(20) not null));
```

Insert syntax is similar to named row types:

```
  insert into class values (300, “Performance and Tuning”, “Covers advanced information on tuning the
       Informix Dynamic Server”, (SET{“RDD”,”BSQL”}));
```

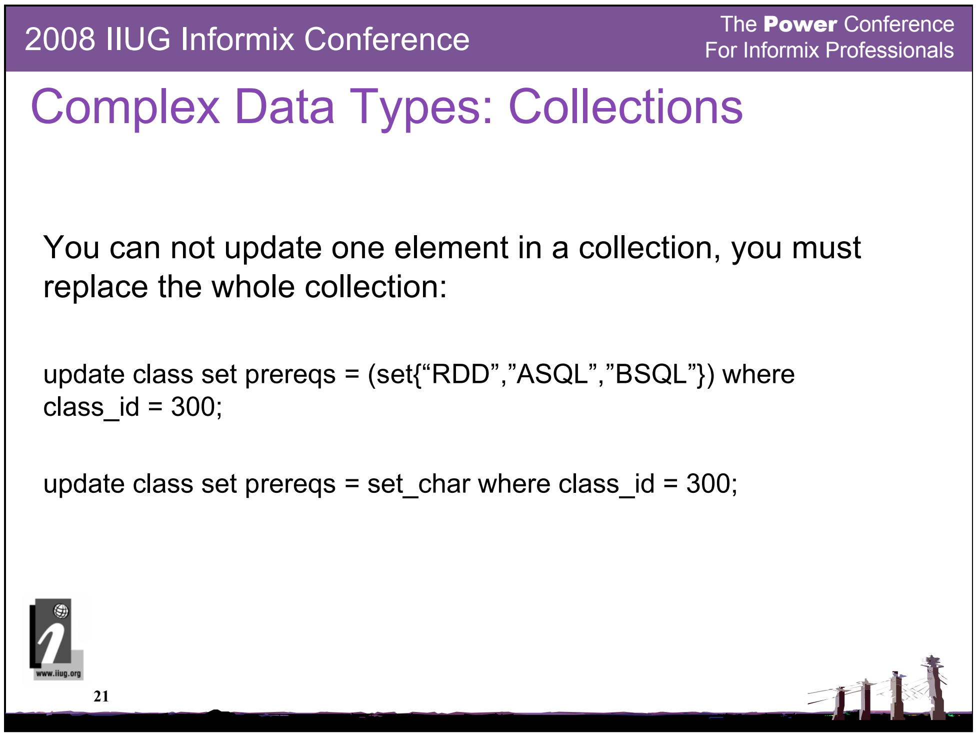
Use the “*in*” keyword to query values in a collection

SQL

```
  select * from class where (“ASQL”) in prereqs;
```

4GL

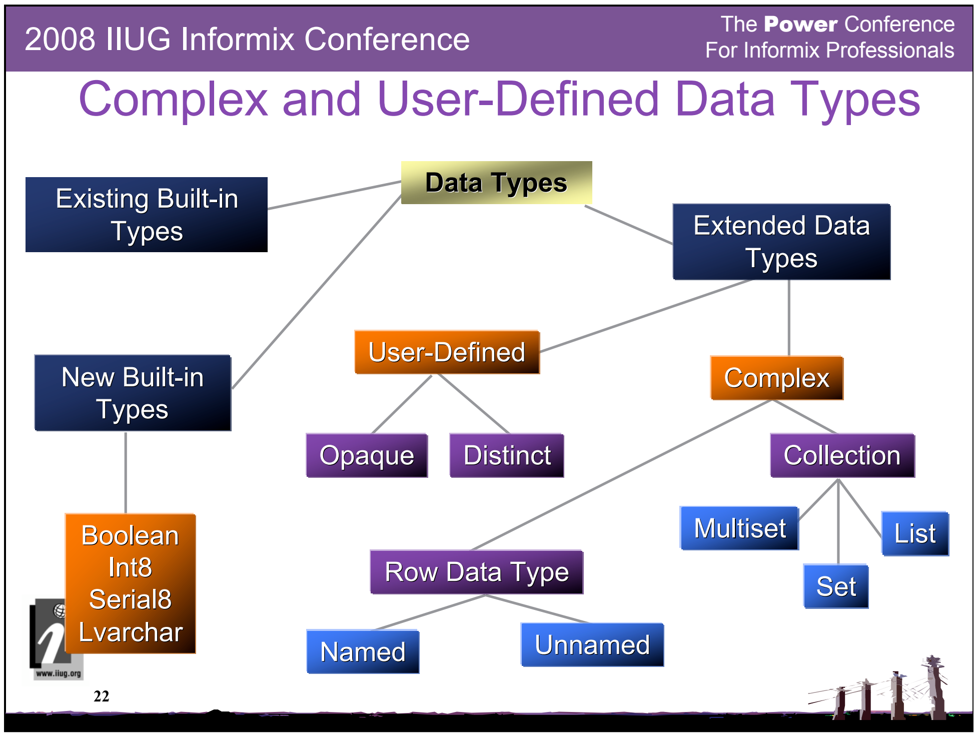
```
  define xyz char(20)
  define set_var set(char(20))
  select prereqs into set_var from class where class_id = 300

  foreach del_set_cursor for
          select * into xyz from table(set_var)
          if xyz matches “RDD” then
                  delete from table(set_var) where current of del_set_cursor
          end if
  end foreach
```



Complex Data Types: Collections

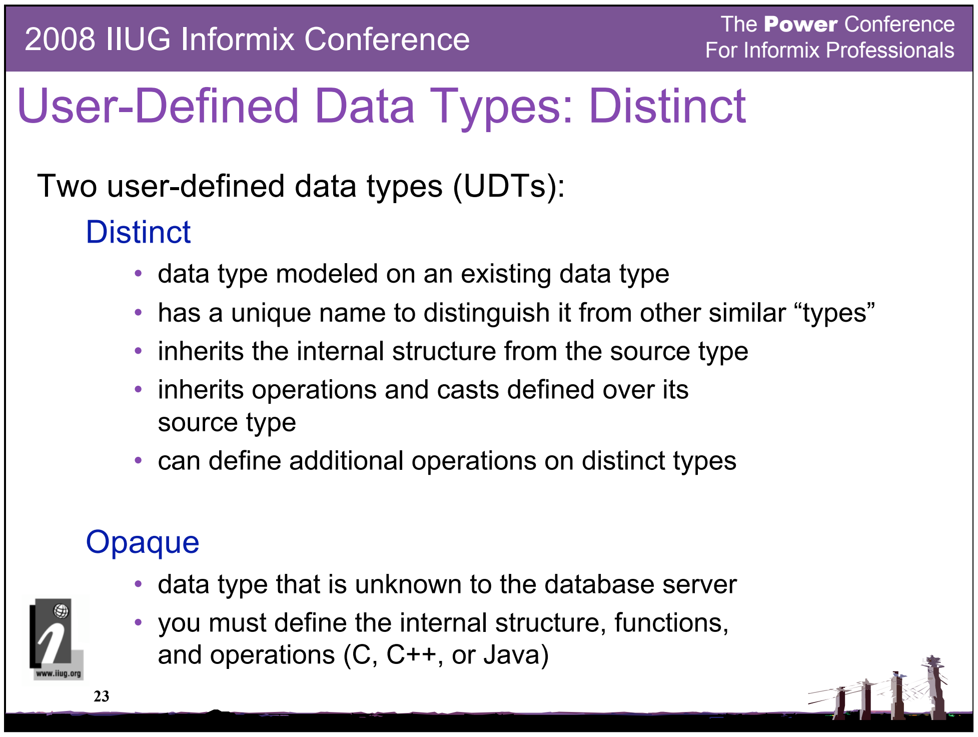
You can not update one element in a collection, you must replace the whole collection:

```
update class set prereqs = (set{“RDD”,”ASQL”,”BSQL”}) where
class_id = 300;

update class set prereqs = set_char where class_id = 300;
```



Complex and User-Defined Data Types

22



User-Defined Data Types: Distinct

Two user-defined data types (UDTs):

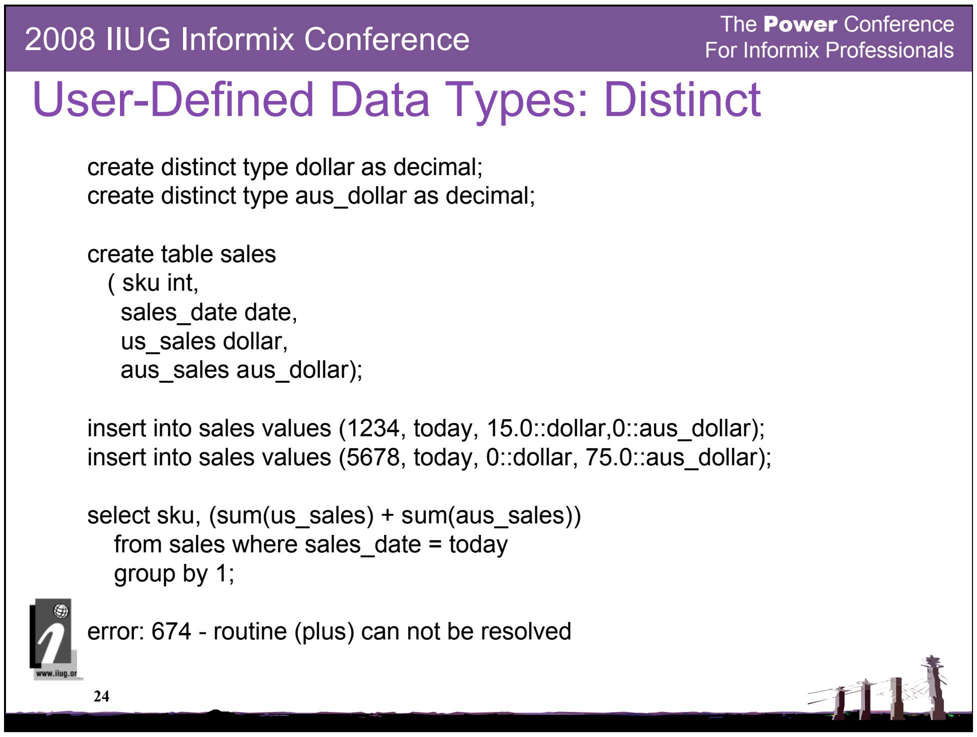
Distinct

- data type modeled on an existing data type
- has a unique name to distinguish it from other similar “types”
- inherits the internal structure from the source type
- inherits operations and casts defined over its source type
- can define additional operations on distinct types

Opaque

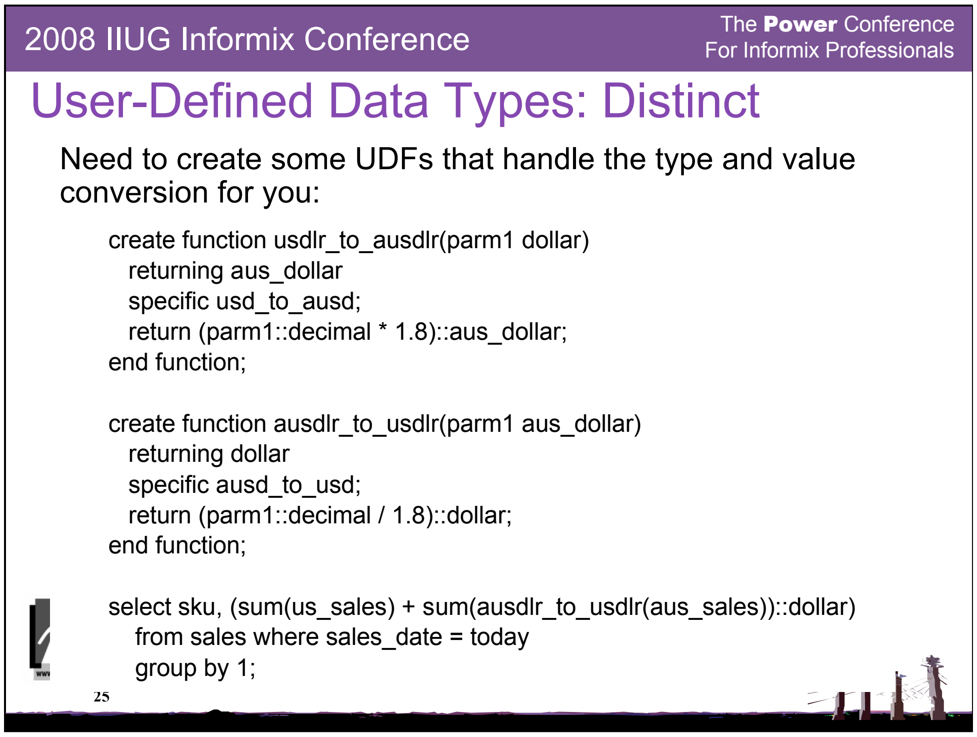
- data type that is unknown to the database server
- you must define the internal structure, functions, and operations (C, C++, or Java)

www.iiug.org



User-Defined Data Types: Distinct

```
create distinct type dollar as decimal;
create distinct type aus_dollar as decimal;

create table sales
   ( sku int,
     sales_date date,
     us_sales dollar,
     aus_sales aus_dollar);

insert into sales values (1234, today, 15.0::dollar,0::aus_dollar);
insert into sales values (5678, today, 0::dollar, 75.0::aus_dollar);

select sku, (sum(us_sales) + sum(aus_sales))
    from sales where sales_date = today
    group by 1;

error: 674 - routine (plus) can not be resolved
```

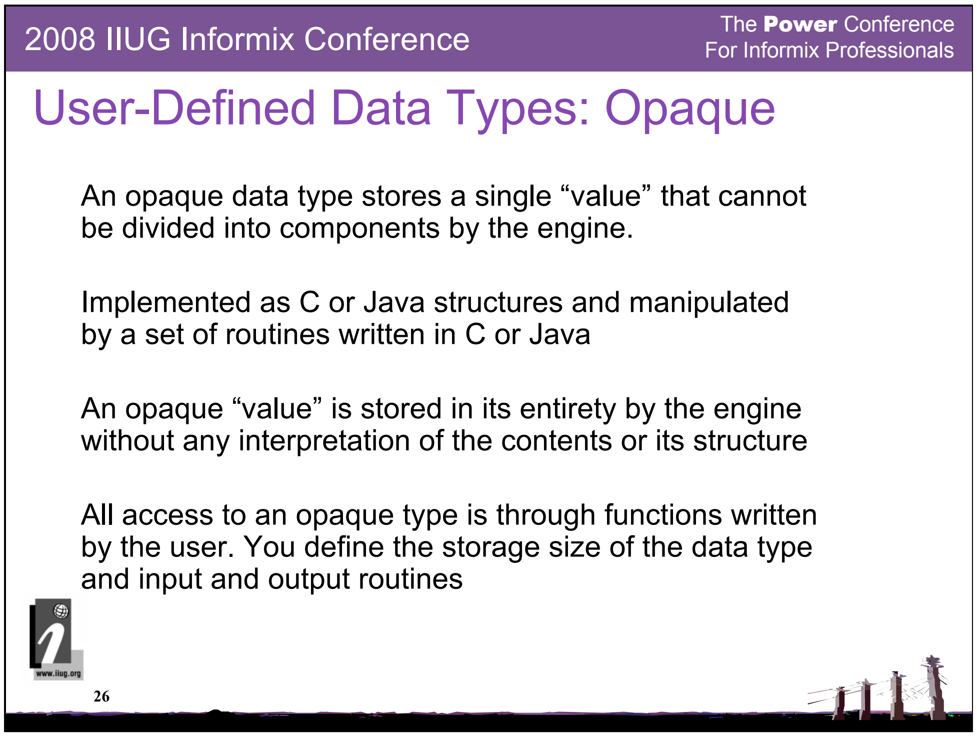


User-Defined Data Types: Distinct

Need to create some UDFs that handle the type and value conversion for you:

```
create function usdlr_to_ausdlr(parm1 dollar)
  returning aus_dollar
  specific usd_to_ausd;
  return (parm1::decimal * 1.8)::aus_dollar;
end function;

create function ausdlr_to_usdlr(parm1 aus_dollar)
  returning dollar
  specific ausd_to_usd;
  return (parm1::decimal / 1.8)::dollar;
end function;

select sku, (sum(us_sales) + sum(ausdlr_to_usdlr(aus_sales))::dollar)
   from sales where sales_date = today
   group by 1;
```

25



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

User-Defined Data Types: Opaque

An opaque data type stores a single “value” that cannot be divided into components by the engine.

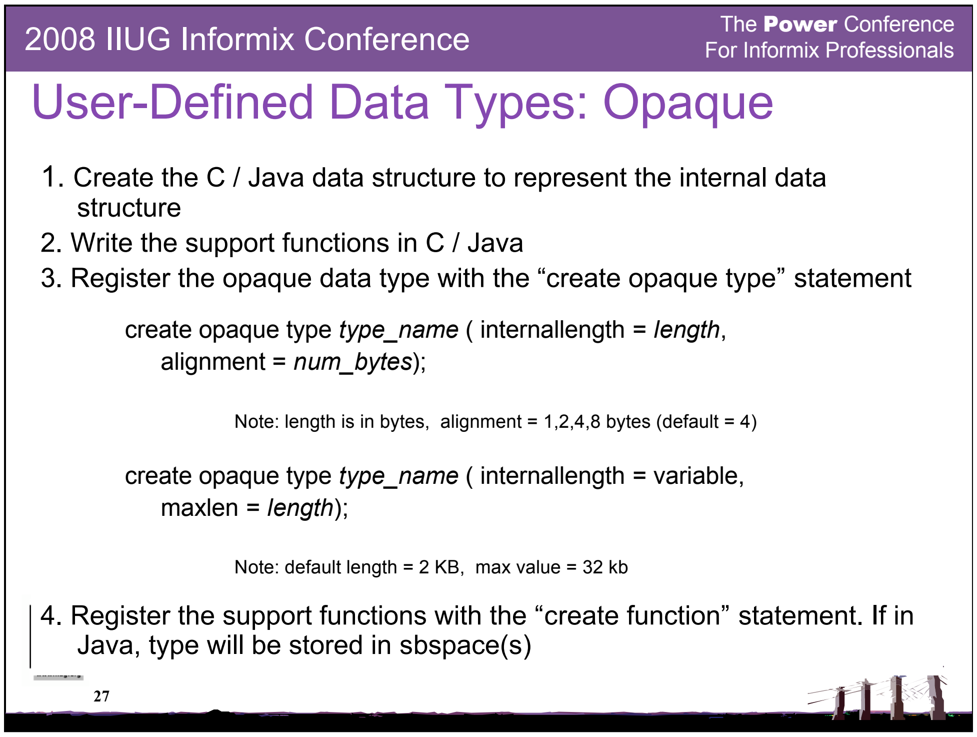
Implemented as C or Java structures and manipulated by a set of routines written in C or Java

An opaque “value” is stored in its entirety by the engine without any interpretation of the contents or its structure

All access to an opaque type is through functions written by the user. You define the storage size of the data type and input and output routines

www.iiug.org

26



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

User-Defined Data Types: Opaque

1. Create the C / Java data structure to represent the internal data structure
2. Write the support functions in C / Java
3. Register the opaque data type with the “create opaque type” statement

   create opaque type *type_name* (internallength = *length*,
   alignment = *num_bytes*);

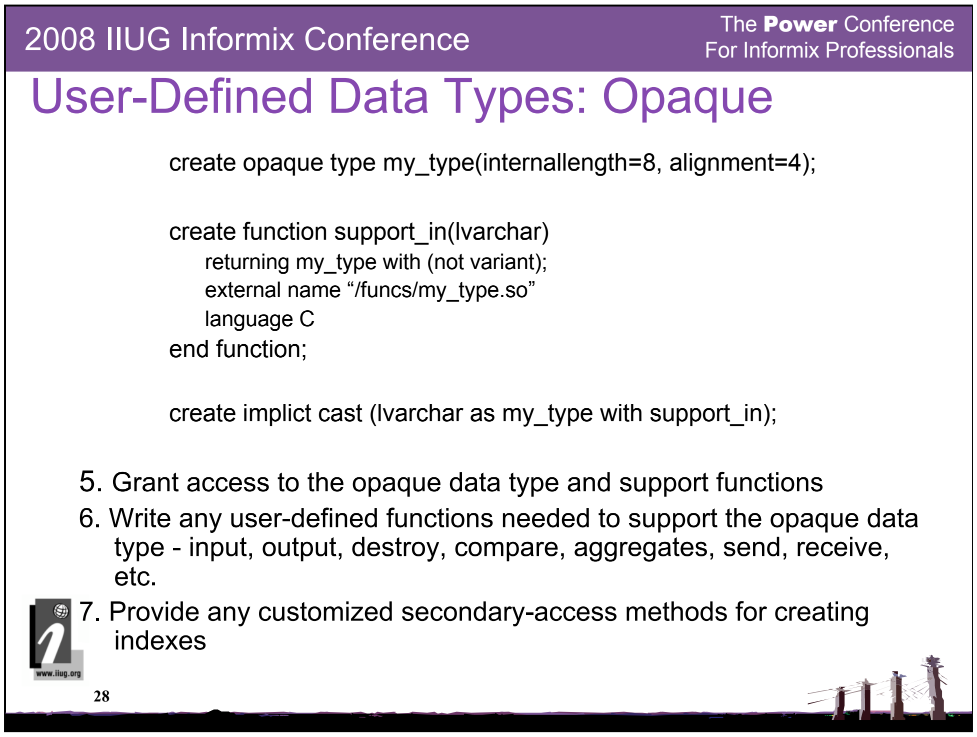
   Note: length is in bytes, alignment = 1,2,4,8 bytes (default = 4)

   create opaque type *type_name* (internallength = variable,
   maxlen = *length*);

   Note: default length = 2 KB, max value = 32 kb

4. Register the support functions with the “create function” statement. If in Java, type will be stored in sbspace(s)

27



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

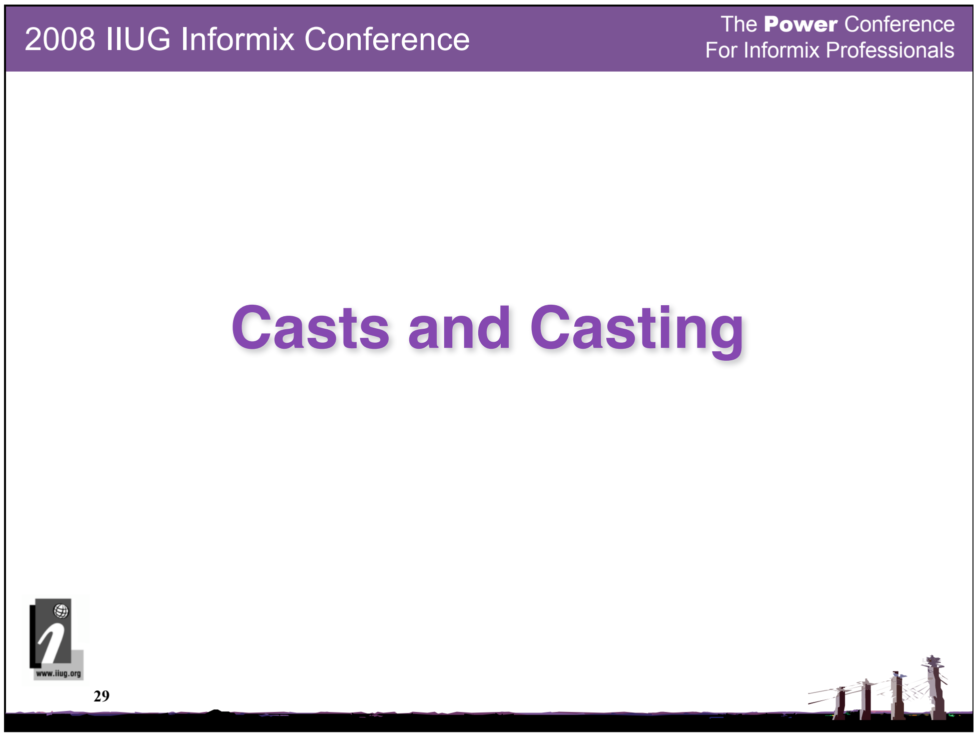
User-Defined Data Types: Opaque

```
create opaque type my_type(internallength=8, alignment=4);

create function support_in(lvarchar)
    returning my_type with (not variant);
    external name “/funcs/my_type.so”
    language C
end function;

create implict cast (lvarchar as my_type with support_in);
```

5. Grant access to the opaque data type and support functions
6. Write any user-defined functions needed to support the opaque data type - input, output, destroy, compare, aggregates, send, receive, etc.
7. Provide any customized secondary-access methods for creating indexes

www.iiug.org

28



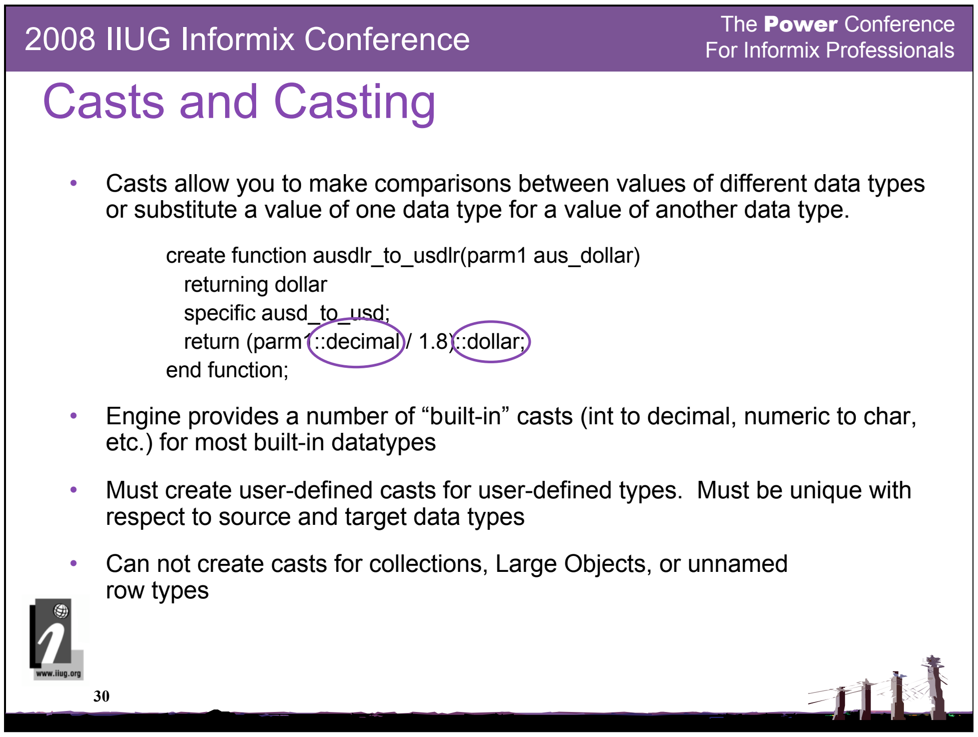
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Casts and Casting

www.iiug.org

29



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Casts and Casting

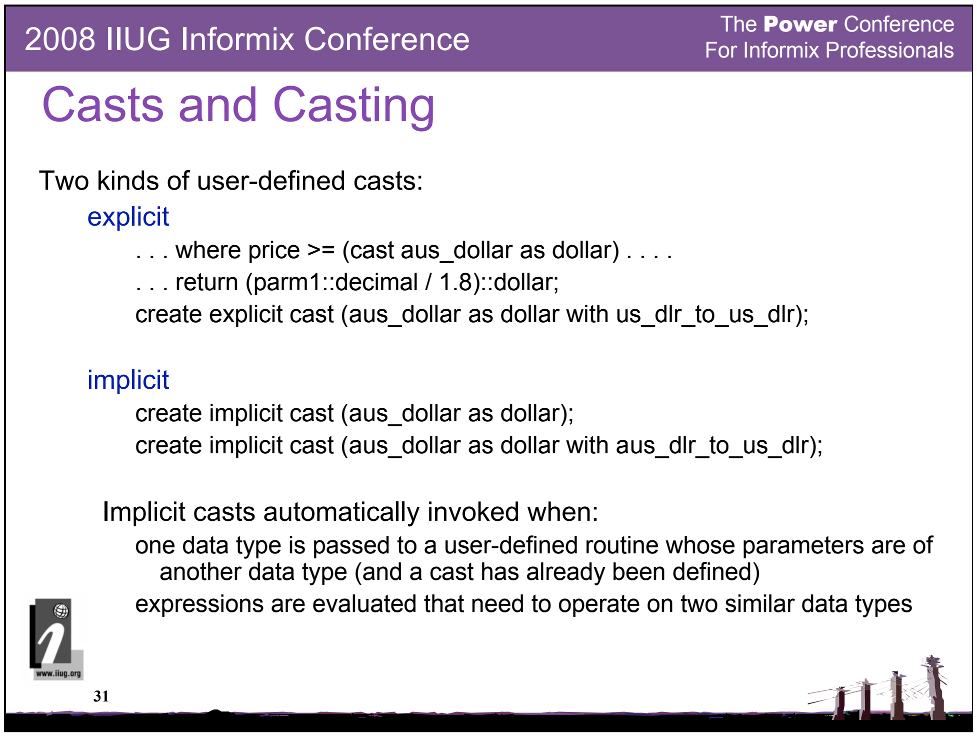
- Casts allow you to make comparisons between values of different data types or substitute a value of one data type for a value of another data type.

```
create function ausdlr_to_usdlr(parm1 aus_dollar)
  returning dollar
  specific ausd_to_usd;
  return (parm1::decimal / 1.8)::dollar;
end function;
```

- Engine provides a number of “built-in” casts (int to decimal, numeric to char, etc.) for most built-in datatypes
- Must create user-defined casts for user-defined types. Must be unique with respect to source and target data types
- Can not create casts for collections, Large Objects, or unnamed row types

www.iiug.org

30



Casts and Casting

Two kinds of user-defined casts:

explicit

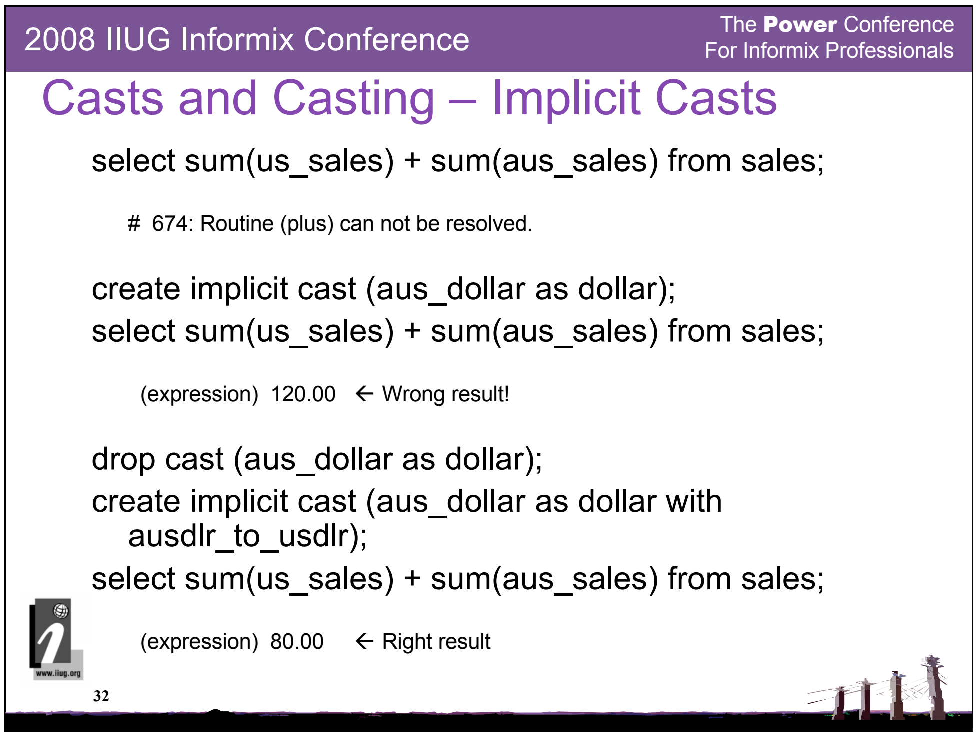
```
. . . where price >= (cast aus_dollar as dollar) . . . .
. . . return (parm1::decimal / 1.8)::dollar;
create explicit cast (aus_dollar as dollar with us_dlr_to_us_dlr);
```

implicit

```
create implicit cast (aus_dollar as dollar);
create implicit cast (aus_dollar as dollar with aus_dlr_to_us_dlr);
```

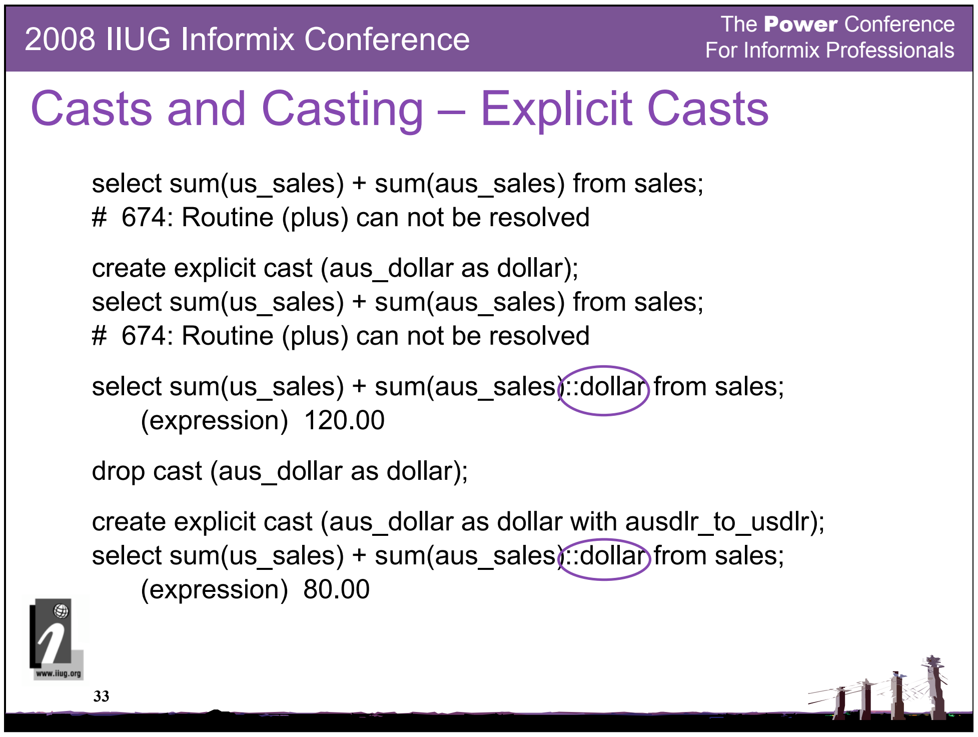
Implicit casts automatically invoked when:

- one data type is passed to a user-defined routine whose parameters are of another data type (and a cast has already been defined)
- expressions are evaluated that need to operate on two similar data types



Casts and Casting – Implicit Casts

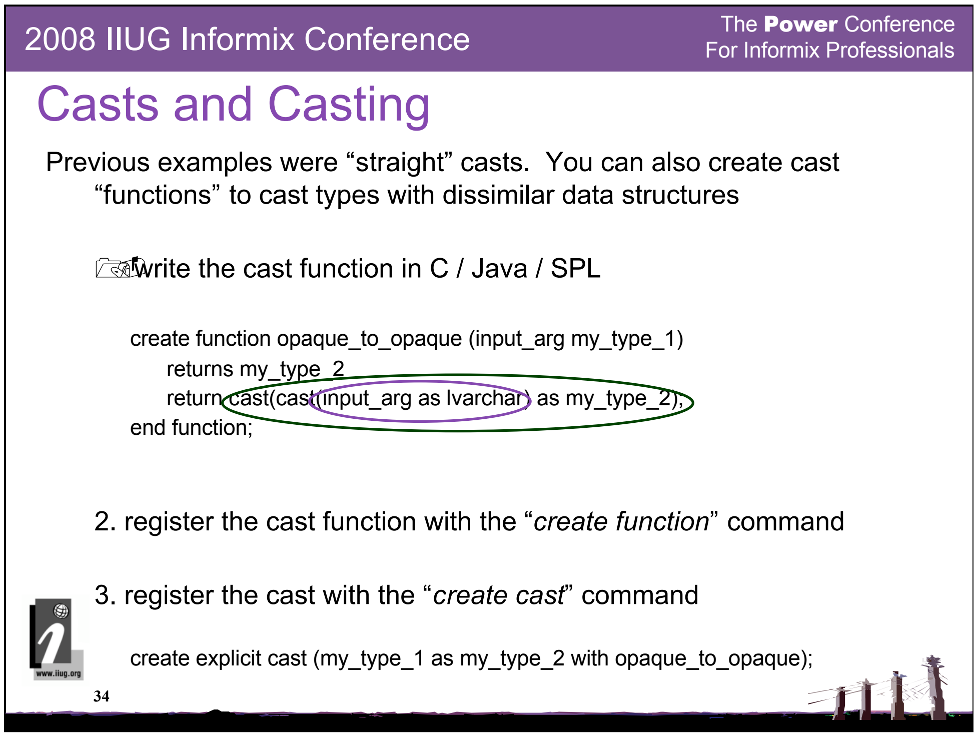
```
select sum(us_sales) + sum(aus_sales) from sales;

   #  674: Routine (plus) can not be resolved.

create implicit cast (aus_dollar as dollar);
select sum(us_sales) + sum(aus_sales) from sales;

    (expression)  120.00    ← Wrong result!

drop cast (aus_dollar as dollar);
create implicit cast (aus_dollar as dollar with
   ausdlr_to_usdlr);
select sum(us_sales) + sum(aus_sales) from sales;

    (expression)  80.00     ← Right result
```



Casts and Casting – Explicit Casts

```
select sum(us_sales) + sum(aus_sales) from sales;
#  674: Routine (plus) can not be resolved

create explicit cast (aus_dollar as dollar);
select sum(us_sales) + sum(aus_sales) from sales;
#  674: Routine (plus) can not be resolved

select sum(us_sales) + sum(aus_sales)::dollar from sales;
    (expression)  120.00

drop cast (aus_dollar as dollar);

create explicit cast (aus_dollar as dollar with ausdlr_to_usdlr);
select sum(us_sales) + sum(aus_sales)::dollar from sales;
    (expression)  80.00
```

www.iiug.org



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Casts and Casting

Previous examples were "straight" casts. You can also create cast "functions" to cast types with dissimilar data structures

☞ Write the cast function in C / Java / SPL

```
create function opaque_to_opaque (input_arg my_type_1)
    returns my_type_2
    return cast(cast(input_arg as lvarchar) as my_type_2);
end function;
```

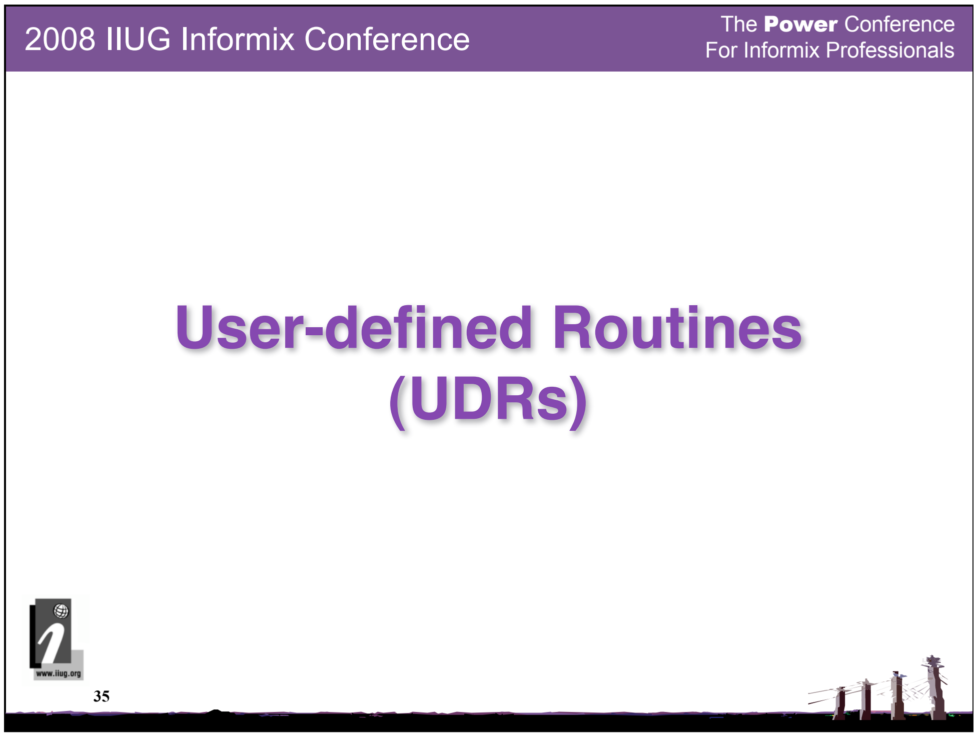
2. register the cast function with the "*create function*" command

3. register the cast with the "*create cast*" command

```
create explicit cast (my_type_1 as my_type_2 with opaque_to_opaque);
```

www.iiug.org

34



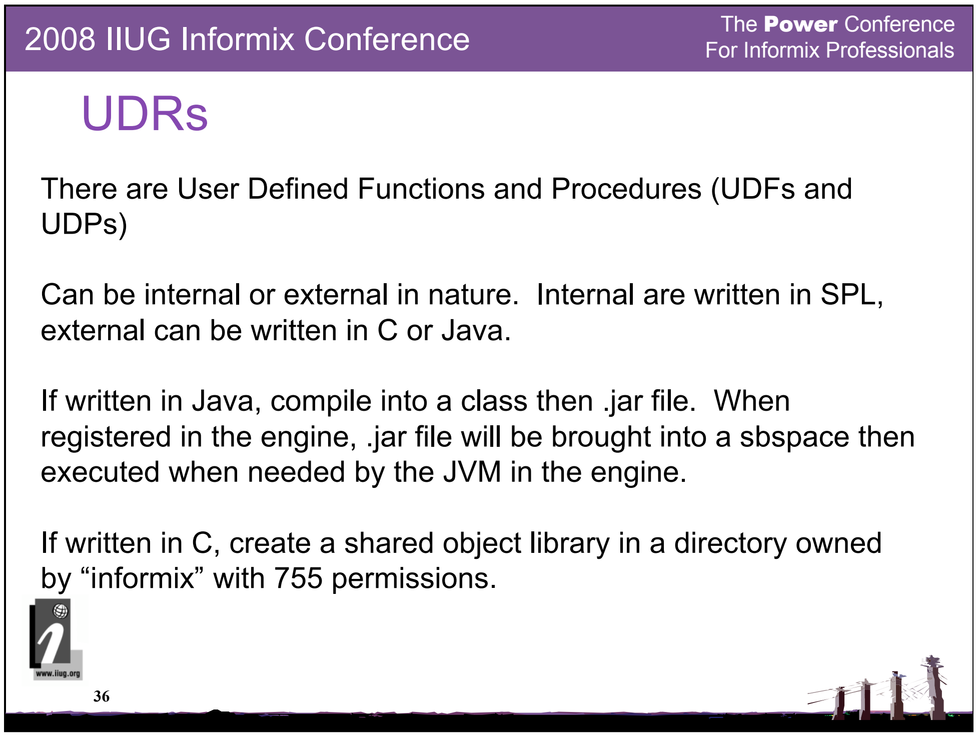
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

User-defined Routines (UDRs)

www.iiug.org

35



UDRs

There are User Defined Functions and Procedures (UDFs and UDPs)

Can be internal or external in nature. Internal are written in SPL, external can be written in C or Java.

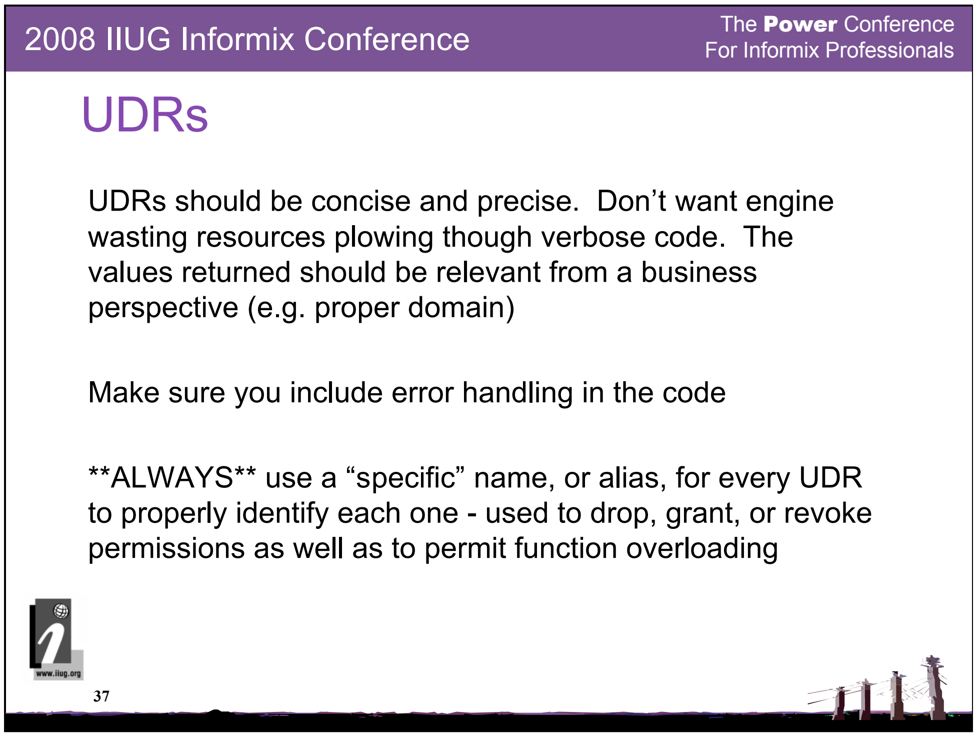
If written in Java, compile into a class then .jar file. When registered in the engine, .jar file will be brought into a sbspace then executed when needed by the JVM in the engine.

If written in C, create a shared object library in a directory owned by “informix” with 755 permissions.

www.iiug.org

36

UDF -



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

UDRs

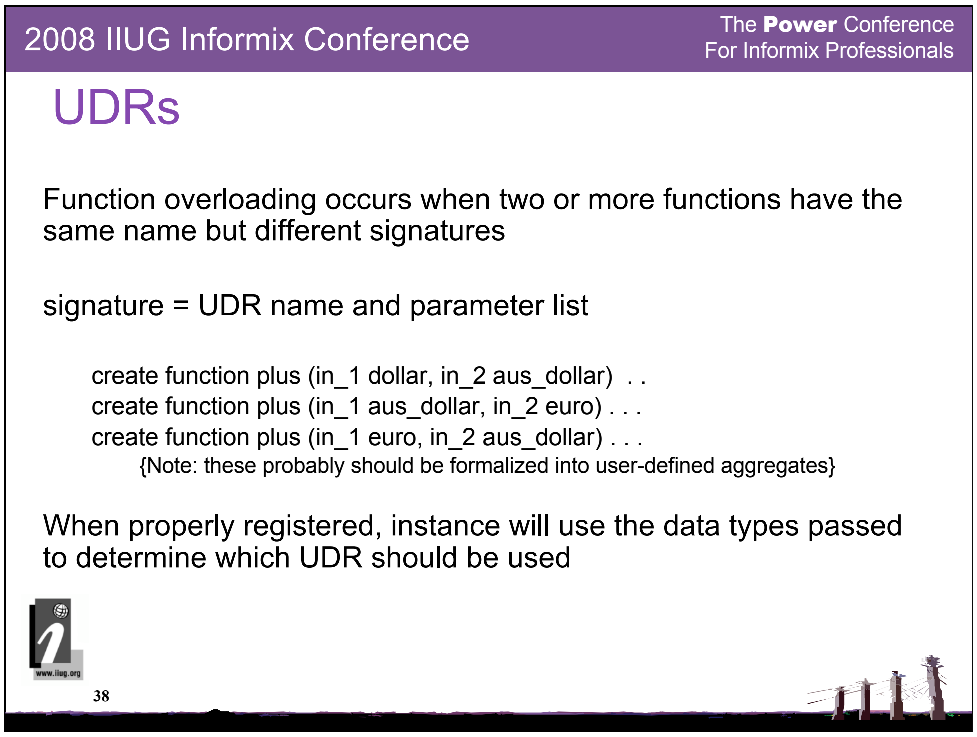
UDRs should be concise and precise. Don’t want engine wasting resources plowing though verbose code. The values returned should be relevant from a business perspective (e.g. proper domain)

Make sure you include error handling in the code

ALWAYS use a “specific” name, or alias, for every UDR to properly identify each one - used to drop, grant, or revoke permissions as well as to permit function overloading

www.iiug.org

37



UDRs

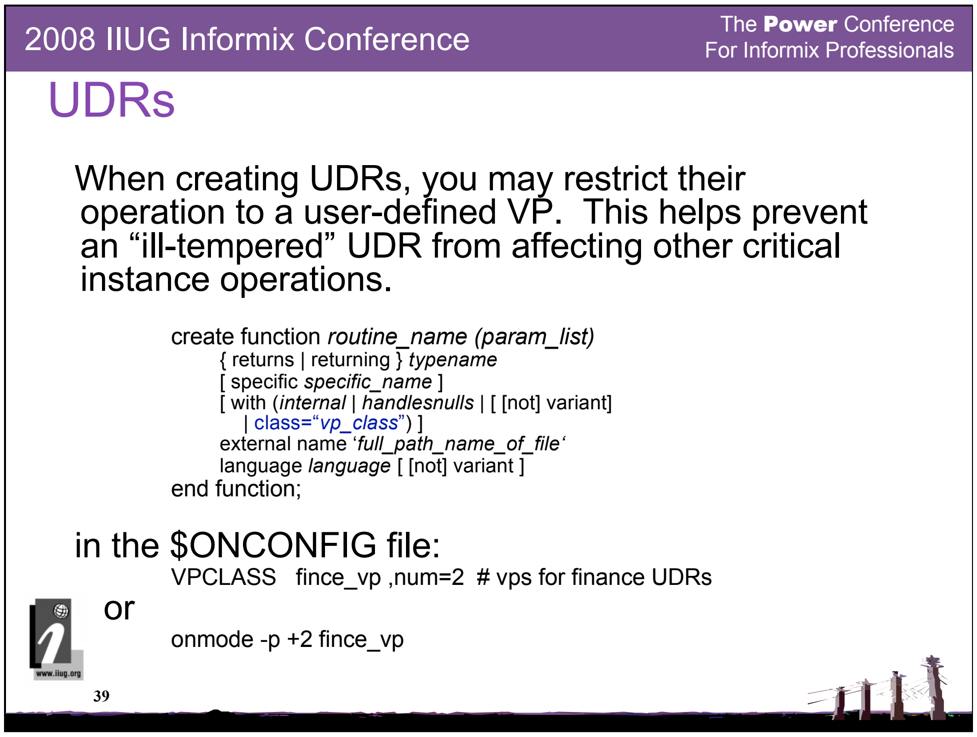
Function overloading occurs when two or more functions have the same name but different signatures

signature = UDR name and parameter list

```
create function plus (in_1 dollar, in_2 aus_dollar)  . .
create function plus (in_1 aus_dollar, in_2 euro) . . .
create function plus (in_1 euro, in_2 aus_dollar) . . .
```

{Note: these probably should be formalized into user-defined aggregates}

When properly registered, instance will use the data types passed to determine which UDR should be used



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

UDRs

When creating UDRs, you may restrict their operation to a user-defined VP. This helps prevent an “ill-tempered” UDR from affecting other critical instance operations.

```
create function routine_name (param_list)
        { returns | returning } typename
        [ specific specific_name ]
        [ with (internal | handlesnulls | [ [not] variant]
            | class=“vp_class”) ]
        external name ‘full_path_name_of_file‘
        language language [ [not] variant ]
end function;
```

in the $ONCONFIG file:

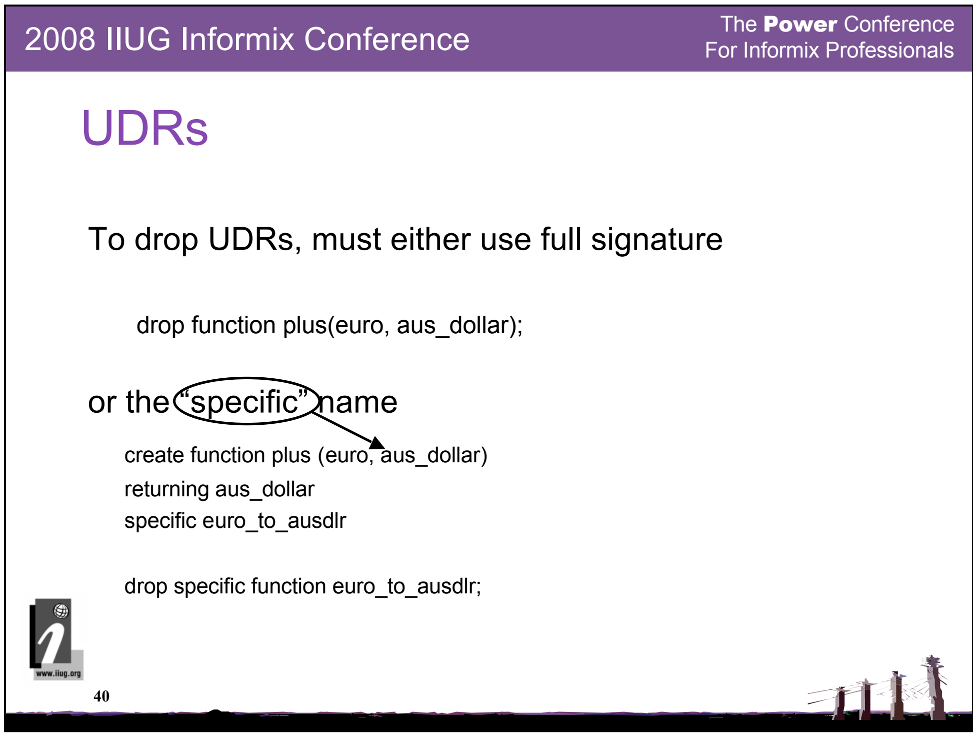
```
VPCLASS   fince_vp ,num=2  # vps for finance UDRs
```

or

```
onmode -p +2 fince_vp
```

www.iiug.org

39



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

UDRs

To drop UDRs, must either use full signature

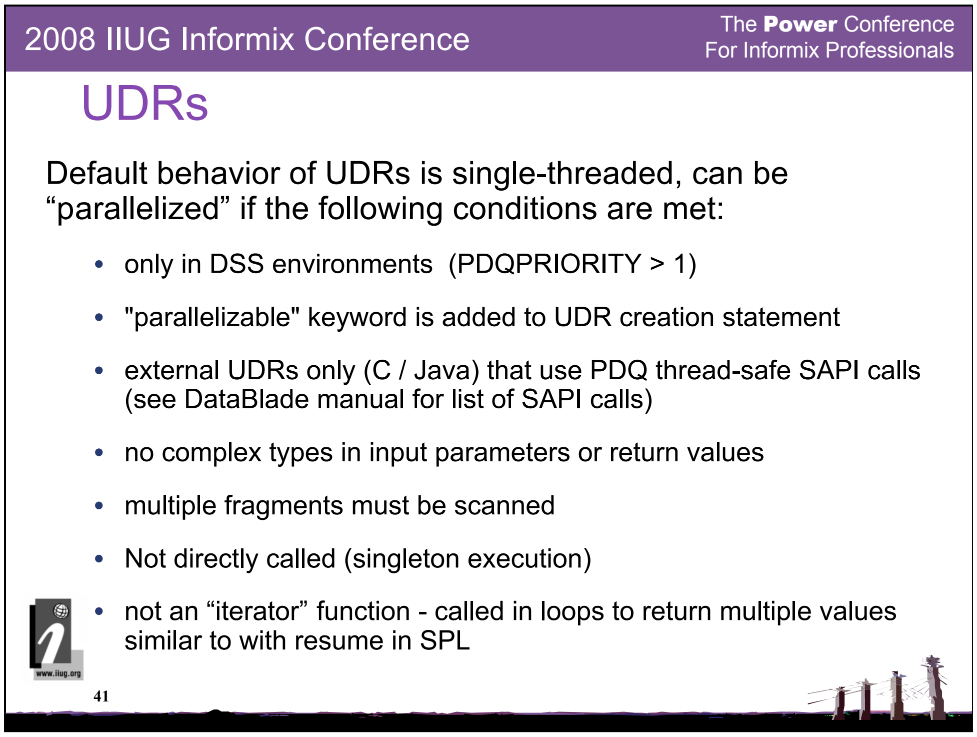
```
drop function plus(euro, aus_dollar);
```

or the “specific” name

```
create function plus (euro, aus_dollar)
returning aus_dollar
specific euro_to_ausdlr

drop specific function euro_to_ausdlr;
```

www.iiug.org

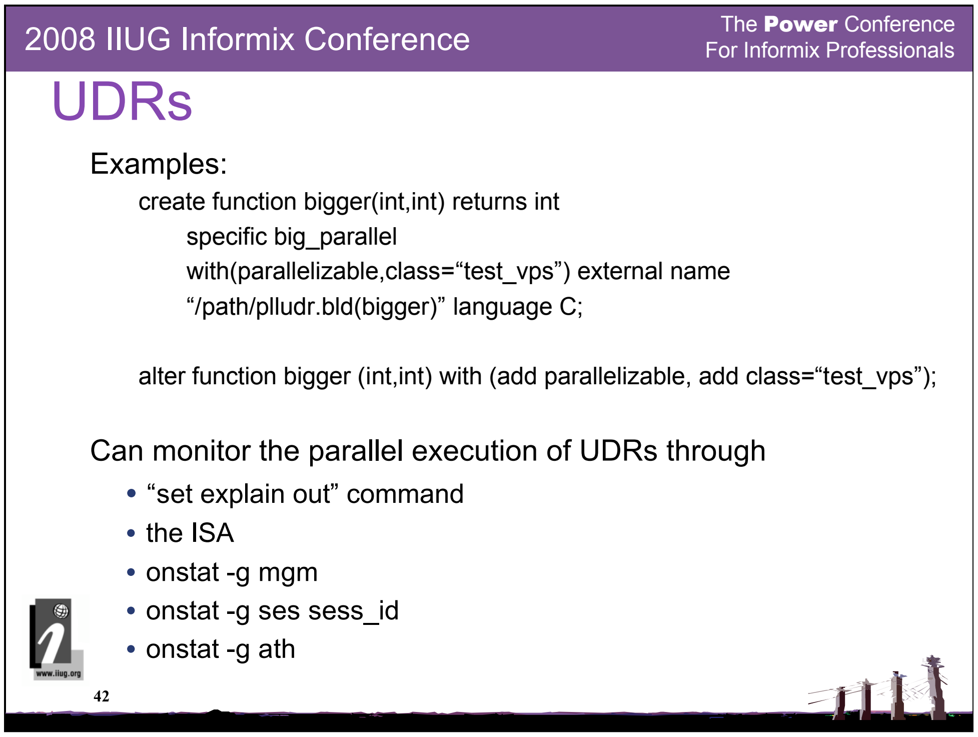
40



UDRs

Default behavior of UDRs is single-threaded, can be "parallelized" if the following conditions are met:

- only in DSS environments (PDQPRIORITY > 1)
- "parallelizable" keyword is added to UDR creation statement
- external UDRs only (C / Java) that use PDQ thread-safe SAPI calls (see DataBlade manual for list of SAPI calls)
- no complex types in input parameters or return values
- multiple fragments must be scanned
- Not directly called (singleton execution)
- not an "iterator" function - called in loops to return multiple values similar to with resume in SPL



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

UDRs

Examples:

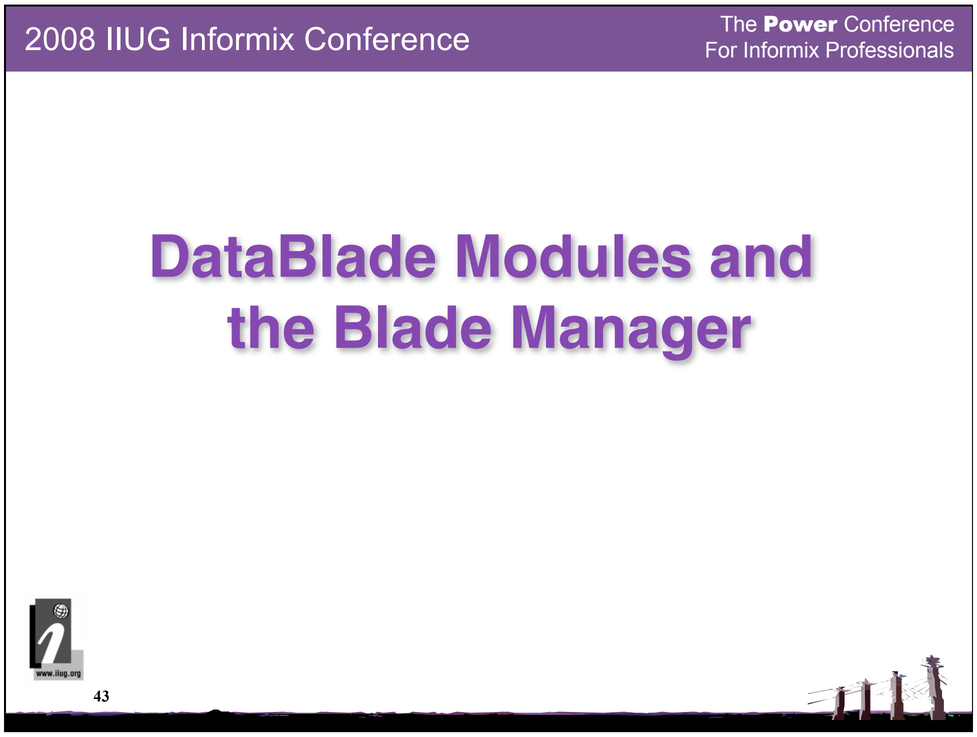
```
create function bigger(int,int) returns int
    specific big_parallel
    with(parallelizable,class="test_vps") external name
    "/path/plludr.bld(bigger)" language C;

alter function bigger (int,int) with (add parallelizable, add class="test_vps");
```

Can monitor the parallel execution of UDRs through

- "set explain out" command
- the ISA
- onstat -g mgm
- onstat -g ses sess_id
- onstat -g ath

www.iiug.org

42



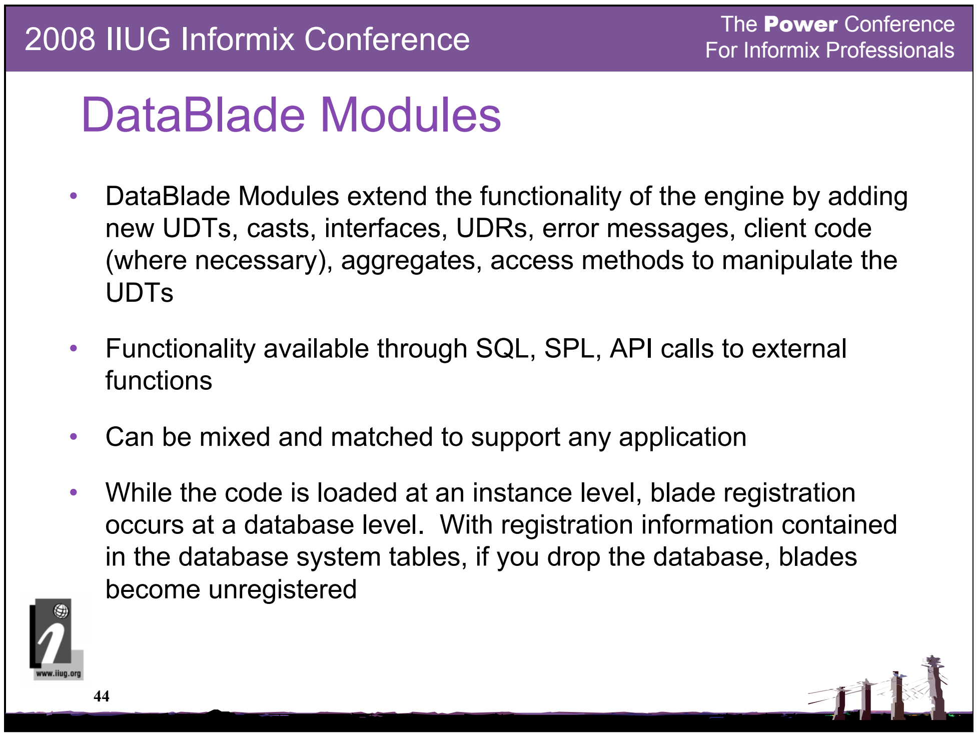
2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

DataBlade Modules and the Blade Manager

www.iiug.org

43

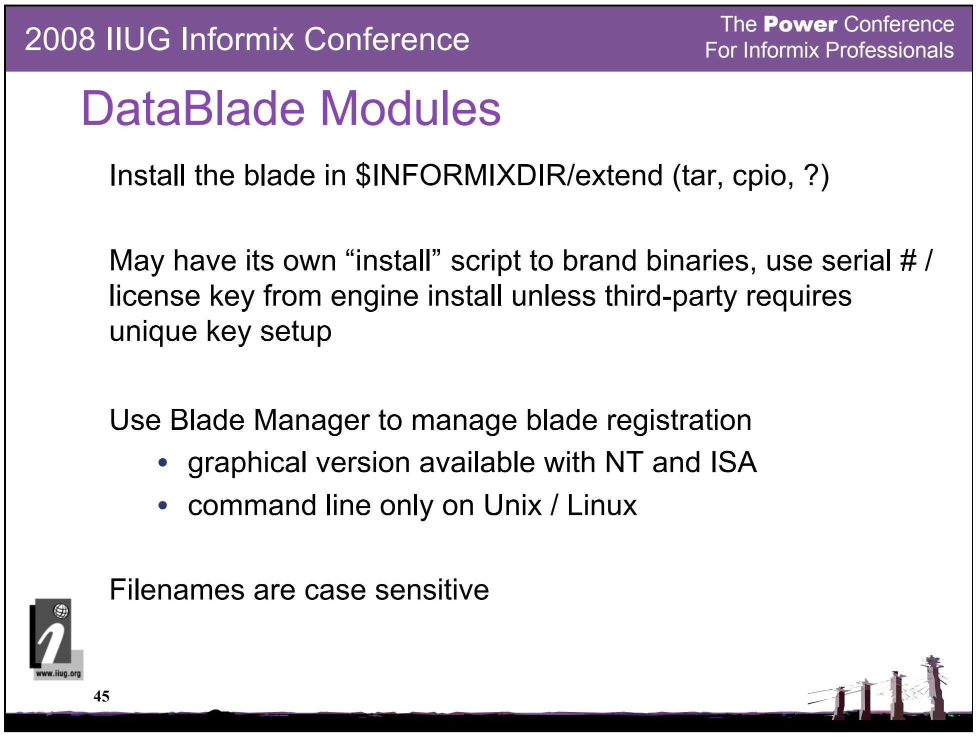


DataBlade Modules

- DataBlade Modules extend the functionality of the engine by adding new UDTs, casts, interfaces, UDRs, error messages, client code (where necessary), aggregates, access methods to manipulate the UDTs
- Functionality available through SQL, SPL, API calls to external functions
- Can be mixed and matched to support any application
- While the code is loaded at an instance level, blade registration occurs at a database level. With registration information contained in the database system tables, if you drop the database, blades become unregistered

www.iiug.org

44



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

DataBlade Modules

Install the blade in $INFORMIXDIR/extend (tar, cpio, ?)

May have its own “install” script to brand binaries, use serial # / license key from engine install unless third-party requires unique key setup

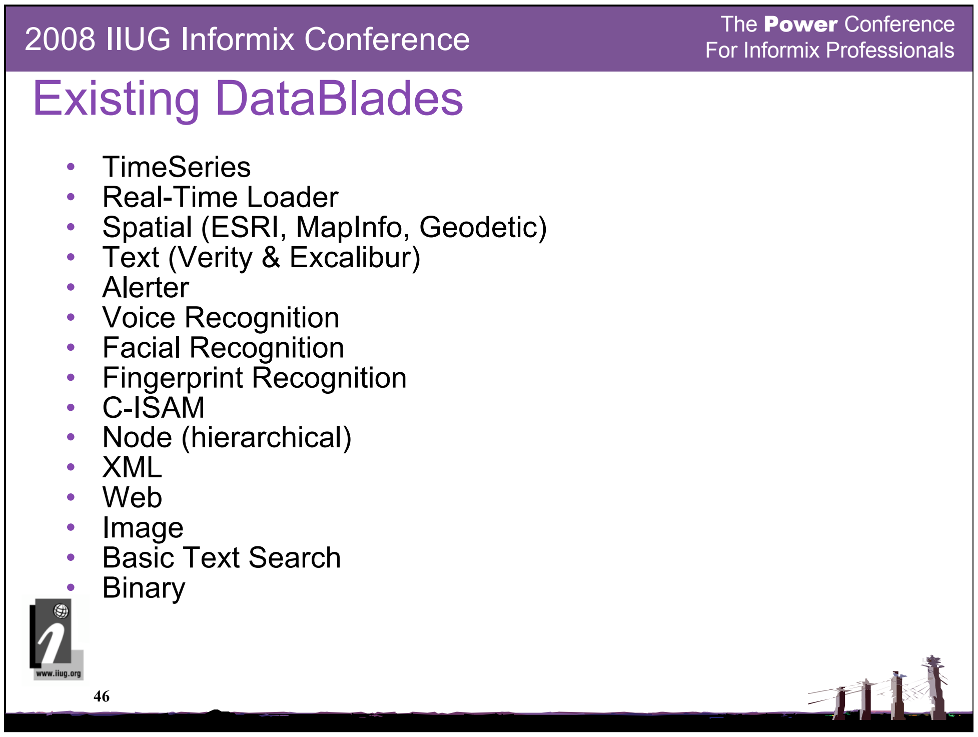
Use Blade Manager to manage blade registration

- graphical version available with NT and ISA
- command line only on Unix / Linux

Filenames are case sensitive

www.iiug.org

45



2008 IIUG Informix Conference

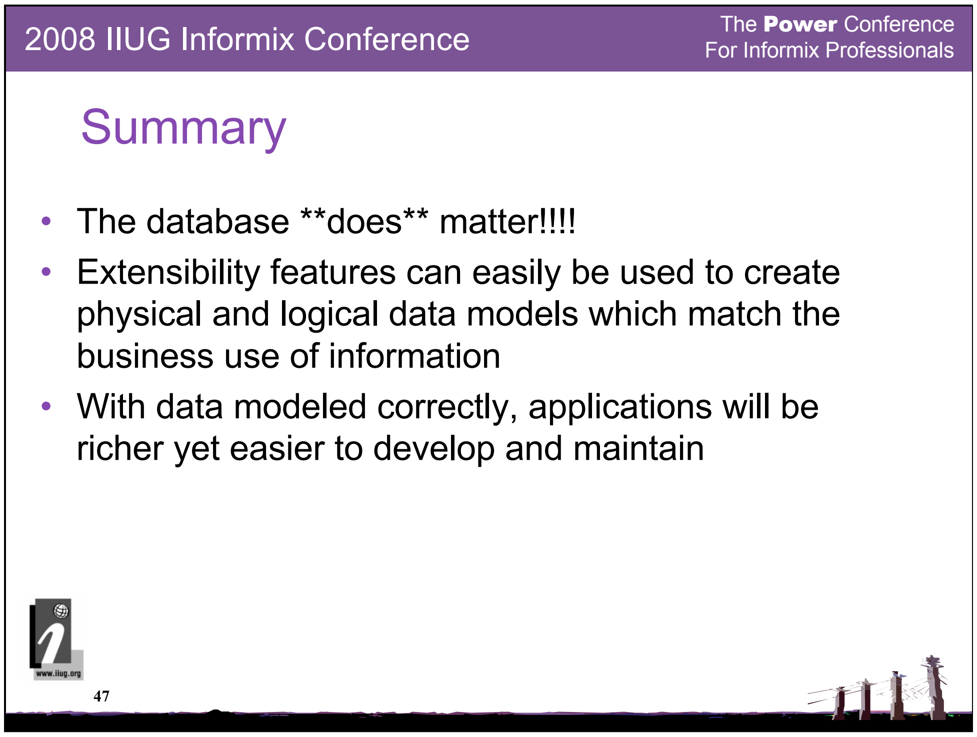
The **Power** Conference
For Informix Professionals

Existing DataBlades

- TimeSeries
- Real-Time Loader
- Spatial (ESRI, MapInfo, Geodetic)
- Text (Verity & Excalibur)
- Alerter
- Voice Recognition
- Facial Recognition
- Fingerprint Recognition
- C-ISAM
- Node (hierarchical)
- XML
- Web
- Image
- Basic Text Search
- Binary

www.iiug.org

46



2008 IIUG Informix Conference

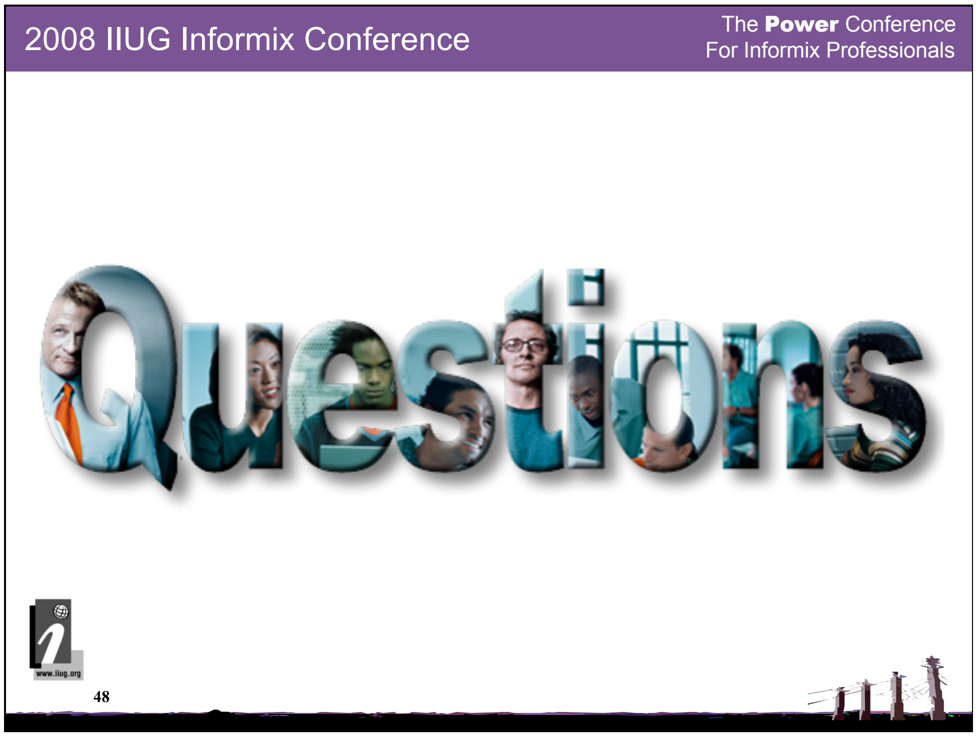
The **Power** Conference
For Informix Professionals

Summary

- The database **does** matter!!!!
- Extensibility features can easily be used to create physical and logical data models which match the business use of information
- With data modeled correctly, applications will be richer yet easier to develop and maintain

www.iiug.org

47



2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

www.iiug.org

48

2008 IIUG Informix Conference

The **Power** Conference
For Informix Professionals

Thank You

www.iiug.org

49