Uvis Reference Card v2.4

Text in gray: Maybe implement later

Paths in tables - Join paths

Rows: Map.tblA Get tblA from the data map. Generate a component for each data row.

Rows: tblA Where zip = 35
Get only the rows where the zip field is 35.

tblA Where zip = txtCrit!Text
Get only the rows where the zip field matches the text in txtCrit.

tblA Where name Like txtCrit!Text & "%"
Wildcarding: Get only the rows where the name field matches the text in txtCrit followed by any characters (%).

tblA-<relB **(Left) join many**
Get all B rows that relate to tblA. (Usually there are more B rows than A rows.) Add the fields from tblA to each row. If a tblA row has no related B rows, include a null row and add the fields from tblA to it.

tblA=<relB **Inner join many**
As left join many, but omit tblA rows without a related B row.

tblA>-relB **(Left) join one**
Get all tblA rows, add fields from the related B row. If there is no related B row, add null fields.

tblA>=relB **Inner join one**
As left join one, but omit A rows without a related B row. (*Inner join one* is not allowed if the path has an earlier left join.)

SQL tails

A join path can have an SQL tail with optional Where, Group By, Having and Order By parts, in this sequence. Can also have optional Top and Create parts.

Note: Joins in Uvis have no Select part. Uvis decides what to select based on the property formulas.

tblA **Where** id=2 And time > CDate(txtCrit !Text)
Get only the rows where id=2 and time > a dialog value.

tblA -< relB **Where** tblA.id=2 And relB.code>txtCrit !Text
Get rows based on fields in tblA as well as the related B.

tblA **Where** name>"ab" **Order By** name **Desc,** id
Get the rows where ... Order them by name in descending order and for those with the same name, id in ascending order.

tblA **Group By** zip
Divide all tblA rows into groups according to their zip. Calculate aggregates for each group according to formulas using Min(x), Count(x), etc. The result has one row for each group.

tblA -< relB **Where** relB.name>"ab" **Group By** tblA.zip
Get all B rows related to the A rows and include the A fields. Include only rows with name>"ab". Divide into groups according to tblA's zip field.

tblA -< relB **Group By** tblA.zip **Having** Min(id)>12
Group by zip, but include only groups where the smallest id is larger than 12.

tblA Where time>txtCrit !Text **Top** 10 **Create** 2
Include only the first 10 rows and add 2 empty ones.

tblA Where @ Form ! Include
Insert the text from Form ! Include into the SQL-string

Statements - event handling

Event handler formulas. Done when event triggered.

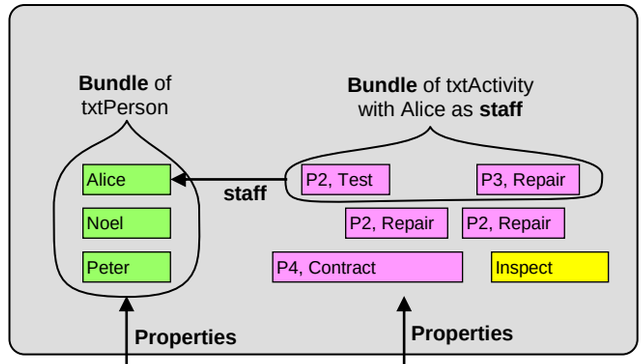
OpenForm("F", 1, "ab")	Open form F with two params. Close any other open F.
OpenMulti("F", 2)	Open F, keep clones of F.
CloseForm(c)	Close c's form.
CloseMulti("F")	Close all clones of form F.
SetFocus(c)	Let c receive all keyboard events.
GotFocus(c)	True when c receives key events.
RefreshForm("F")	Recalculate all form F instances; requery where formula changed.
RefreshForm(c)	Recalculate c's form.
Refresh()	Recalculate all open forms.
RequeryForm("F")	Requery and recalculate.
RequeryForm(c)	Requery and recalculate.
Requery()	Requery and recalculate all.
MsgBox("Go?", 0)	Show a message box and wait for the user to click OK.
ShowPopup(s, c)	Show the popup text s next to c.
ClosePopup(c)	Close the popup next to c.
MsgLog("Done")	Write the message to the log.
c ! Top=3	Set c's Top value to 3.
c.addr = txtAddr!Text	Set the addr field of c's data row to the text in txtAddr.
CreateRow(c)	Create an empty row in c's primary table in memory.
DeleteRow(c[0])	Delete the first of c's rows.
CommitForm(c)	Save all row changes in c's Form in the database.
CommitOne(c)	Save all changes of c's rows.
IsDirty(c)	True if unsaved changes in c's form
CancelUpdates(c)	Cancel changes in c's rows.
if (b) OpenForm(...)	When b is true, open the form.
if (b) { A; B} else { C; D}	When b is true do A and B, else do C and D.
for (i=0; i<10; i=i+1) OpenMulti("F"+i)	Open F0, F1 ... F9
Return	Return from the event handler.

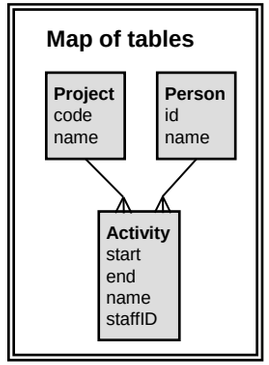
SQL aggregate functions

SQL aggregate functions need a Rows formula with a Group By. Each row links to a group of source rows. SQL aggregate functions count the source rows in the group, sums them, etc.

Null handling: Omits rows where the field is null.

Me.Count(x)	The number of rows in my group, omitting rows with field x = null.
.Sum(x)	The sum of the x fields in my group.
Min(x)	The minimum of the x fields in ...
Max(x)	The maximum of the x fields in ...
Avg(x)	The average of the x fields in ...
Var(x)	Variance of x fields (divide by Count-1)
VarP(x)	Variance (divide by Count)
StDev(x)	Standard deviation (square root of Var).
StDevP(x)	Standard deviation (square root of VarP)
First(x)	The first non-null x field that the dabase engine happens to find in the group.





Template for TextBox: txtPerson

Rows:	Map.Person
Top:	10 + Index * 25
Left::	10
Width:	100
Text:	Person.name
BackColor:	Color.LightGreen
Click:	Requery( )

Template for TextBox: txtActivity

Rows:	Activity >- Project Where start > #1-1-2017#
Top:	staff ! Top
Left:	(start - #1-1-2017#) * 2.5 ' 2.5 pixels per day
Right:	(end - #1-1-2017#) * 2.5
Text:	Project.code & ", " & Activity.name Default staff.name
BackColor:	Project.code=null ? Color.Yellow : Color.Purple
Staff:	Find txtPerson On Person.id = Me.staffID Order By Me.start

Paths from the Me component

Prefixes (Me, Map . . .) can be omitted if unambiguous. The first dot or bang can be omitted if unambiguous.

Me.id	**.** (d**ot):** Field in my data row.
.id id	Omitting prefixes.
tblA.id	**Table prefix:** Walk to the tblA part of my row. Get its id Field.
Me!Top	! (b**ang):** Property in my component.
!Top Top	Omitting prefixes.
!HPos(t)	**Function:** Call property function.
Index	My number in my bundle.
Me[Index-1]	**Indexing:** The previous component in my bundle.
Me[0]	The first component in my bundle.
Me[Last]	The last component in my bundle.
Form !Top	**Form:** Walk to my form. Get its Top.
Param[1]	**Param:** The second form parameter.
Me!txtID ! Top	**Component walk:** Walk to my related txtID box. Get its Top property.
Me!txtID[Last] !Top	Walk to my related txtID box. Get the last in its bundle. Get its Top property.

Find-bundle Me is "txtActivity":
Staff: Find txtPerson On Person.id = Me.staffID Order By Me.start
Find the txtPerson component with id = my staffID. Bundle us having the same txtPerson. Order us by our start time.

Staff ! Top	**Indirect reference:** Walk to my staff component. Get its Top.
heading: Find txtCaption	All of us refer to txtCaption

Path prefixes

Prefixes can resolve any ambiguity. May be omitted if there is no ambiguity.

Me.	Start in my data row.
Me!	Start in my properties.
Main	Start in the component that has my data.
Form	Start in my form component.
Forms	Start in the collection of all forms.
Map	Start in the map of tables.
System	Start in the collection of system items, e.g. System.MouseX or System.OpenForm(…).
Templates	Start in the collection of form templates.

Paths to another Form

Forms ! frmF ! Top **Walk to a form:** From the open forms, get frmF. Get its Top value.

Forms ! frmF(1)!Top **Form bundle:** There may be several open bundles of frmF. Get the second bundle. Get the Top of the first form in this bundle.

Forms!frmF(1)[1]!Top **Form instance:** The second open bundle may contain several open forms. Get the second form. Get its Top.

Paths to Find-bundles

Me is "txtActivity":
Me ! staff ! Bundle[Last] Walk to the bundle of components referring to my staff, walk to the last component of this bundle.
Me ! Staff ! Bundle ! Index My index in this bundle.

Me is "txtPerson":
Me ! Bundle(txtActivity ! staff) [Last] Find the bundle of txtActivity components referring to me through their staff property. Walk to the last component of the bundle.

Text in gray: Maybe implement later

Line continuation and comments

```
Top + index*20          ' Comment

' Comment before line
"long text A" &  _      ' Line continuation and comment
    "long text B"       ' Comment in last line too
```

Constants

23, -23, 0, -4.9E-20	**Decimal numbers**
&h09A0FF, &o177	**Hex and Octal**
Color.Red	**Predefined colors**
Keys.Enter, Font.Arial	**Keyboard keys, fonts, etc.**
"Letter to:"	**Strings**
Chr(65)	The text "A"
"John" & NewLine() "Doe"	Two lines
"Don't say ""no"" "	Don't say "no"
True, False	**Booleans**
Null	**Null and DBnull**
In local format	**Date/time**
#24-12-2011#	24th Dec 2011
#24-12-11 14:15:00#	24th Dec 02 at 14:15

Operators, decreasing precedence

Nulls: Null operands give Null results and error log.

^	Exponentiation
-	Unary minus, 2*-3 = -6
*	Multiply, Result type is Integer, Double, etc.
/	Divide, Single or Double result, 5/2 = 2.5
\	Integer divide, result truncated, 5\3 = 1
Mod	Modulus (remainder), 5 Mod 3 = 2
+ -	Add and subtract numbers and DateTime. DateTime as number of days: Now( ) – 0.5
& +	String concatenation, String result
= <> < > <= >=	Equal, unequal, less than, etc.
s Like "s%n"	Wildcard compare. % any char sequence here. _ any char here. [cz] c or z here. [!cz] not c or z here.
Not	Negation. Bit-wise negation for integers
And	Logical And. Bit-wise And of integers
Or	Logical Or. Bit-wise Or of integers
Xor	Exclusive Or. Bitwise on integers
A ? B : C	If A is true, B else C.
A Default B	A, but B if A is null or error. Errors are not logged. Not allowed in SQL; use IsNull(A).
Init B	B when the form is opened. User actions may change the value later.
Partition(22, 0, 100, 10)	= "20:29" Only in SQL
a Between 3 and 9	Only in SQL
a IN (2, 3, 5, 7)	Only in SQL

Iif and Choose

Iif(a=a, b, c)	= b	**Immediate If**
Iif(a<>a, b, c)	= c	
Iif(Null, b, c)	= c	
Choose(2, a, b, c)	= b	**Choose**
Choose(4, a, b, c)	= Null	
Choose(Null, a, b, c)	= Null	

String functions

Nulls: Null operands give Null results and error report.

Chr(65)	= "A", a one-letter string with this ascii character
Asc("AB")	= 65, Ascii code for first character
Len("A_B")	= 3, length of string.
Left("abc", 2)	= "ab", leftmost two characters
Left("abc", 8)	= "abc", as many as available
Right("abc", 2)	= "bc", rightmost two characters
Mid("abcdef", 2, 3)	= "bcd", three chars, chars 2-4
LTrim(" ab ")	= "ab ", leading spaces removed
RTrim(" ab ")	= " ab", trailing spaces removed
Trim(" ab ")	= "ab", leading and trailing removed
Lcase("A-b")	= "a-b", lower case of all letters
Ucase("A-b")	= "A-B", upper case of all letters
Space(5)	= String of 5 spaces
NewLine()	= String of one new line char

Date-time functions

Null parameters: Always give a Null result.

Now( )	= current DateTime (maybe simulated)
Date( )	= current date, 0:00 (simulated)
ToDay()	The same as Date()
Time( )	= current time (since midnight)
TimeOfDay()	The same as Time()
Day(#25-12-2012#)	= 25, the day as Integer
Month(#25-12-2012#)	= 12, the month as Integer
Year(#25-12-2012#)	= 2012, the year as Integer
Weekday(#25-12-2012#)	= 3 (Sunday=0)
Hour(# ... 13:14:15#)	= 13
Minute(# ... 13:14:15#)	= 14
Second(# ... 13:14:15#)	= 15

```
DateAdd("d", 4, #30-12-2012#) = #03-01-2013#
"y"   "m"   "d"   "h"   "n"   "s"
Year, month, day, hour, minute, second.
Timer( )        = Number of seconds since
                  midnight, with fractional seconds.
DateSerial(2002, 12, 25)   = #12/25/2002#
TimeSerial(12, 28, 48)     = 0.52 (Time 12:28:48)
```

Math functions

```
Sqr(x)      Square root of x. Sqr(9) = 3.
Sin(r), Cos(r), Tan(r), Atn(x), Acos(x), Asin(x)
Trigonometric functions. R measured in radian (180
            degrees = π = 3.141592 radian)
            Sin(0) = 0, Sin(3.141592 / 2) = 1.
Pow(x,y)    X to the power of y. Pow(2, 3) = 8.
Log(x, y)   Logarithm of x with base y. Log(8, 2) = 3.
Rnd( )      A random double number between 0 and 1.
RndInt(n,m) A random integer between n and m-1.
Abs(x)      Returns x for x>=0, -x otherwise.
Hex(x)      Returns a string with the hexadecimal
            value of x. Hex(31) = "1f"
Oct(x)      Returns a string with the octal value of x.
            Oct(31) = "37"
Sgn(x)      Returns 1 for x>0, 0 for x=0, -1 for x<0
Int(x)      Rounds x down to nearest integral value
Fix(x)      Rounds x towards zero
```

Format function

Converts a value to a string, based on a format string. Format characters that are not placeholders, are shown as they are. Backslash+character is shown as the character alone, e.g. \d is shown as d.

Numeric placeholders

```
0       Digit, leading and trailing zero okay here
#       Digit, no leading or trailing zero here
.       Decimal point (shown as the local variant)
,       Thousand separator (shown as the local variant)
e- or e+  Exponent or exponent with plus/minus
%       Show number as percent
```

Format(2.3, "00.00")	= "02.30"
Format(2.36, "#0.0")	= "2.4"
Format(0.3, "##.0#")	= ".3"
Format(32448, "(00)00 00")	= "(03)24 48"
Format(32448, "##.#e+0")	= "32.4e+3"
Format(32448, "##.#E-0")	= "32.4E3"
Format(0.5, "#0.0%")	= "50.0%"

```
;       Separator between formats for positive,
        negative, and zero values:
Format(-3, "000;(000);zero")    = "(003)"
```

Predefined numeric formats

g (general), c (currency), f (fixed), p (percent), e (scientific), n (with thousand separator), x (hex)

Date/time placeholders

```
Example:  DT = #3-2-2002 14:07:09# (Sunday)
Format(DT, "yyyy-MM-dd HH:mm:ss")
          = "2002-02-03 14:07:09"
Format(DT, "dddd yy-MMM-d a\t HH:mm")
          = "Sunday 02-Feb-3 at 14:07"
```

yy	Year, two digits "02"
yyyy	Year, four digits "2002"
M	Month, no leading zero "2" (Interpreted as minutes after h)
MM	Month, two digits "02" (Interpreted as minutes after h)
MMM	Month, short text "Feb"
MMMM	Month, full text "February"
d	Day of month, no leading zero "3"
dd	Day of month, two digits "03"
ddd	Day of week, short text "Sun"
dddd	Day of week, full text "Sunday"
H	Hour, no leading zero, 24-hour clock
HH	Hour, two digits, 24-hour clock
h, hh	Hour, 12-hour clock
tt	Show AM or PM here, 12-hour clock only
zz, zzz	Show time zone, +1, -8.30
m	Minutes, no leading zero "7"
mm	Minutes, two digits "07"
s	Seconds, no leading zero "9"
ss	Seconds, two digits "09"
f, ff ...	Fractions of seconds

Predefined date formats

G (local date and/or time), D (long date), d (short date) T, t (local long/short time), u (local GMT sortable) ...

Conversion and test functions

Nulls: Null operands give Null results and error log.
System prefix can be omitted if unambiguous.

```
System.CInt("2.6")      = 3
CInt("2.6")             = 3, omitting prefix.
Round(2.6)              = 3.0000 (Double)
    Rounding down: See Math functions Int, Fix.
CByte("37")             =37. Overflow outside 0..255
CLng("99456")           = 99456
CDbl("-2.6")            = -2.6
CCur(1/3)               =0.3333 (always 4 dec)

CDate("23-10-03")       = #23-10-2003#
                  Uses local settings for input format
CDate(1)                = "31-12-1899"

CStr(23)                = "23". No preceding space.
CStr(#23-10-2003#)      = "23-10-03 00:00:00"
                  Converts to local date format

IsNull(A)       True if A is null. See also operator Default.
IsDate(v)       True when v is a date or a string that can be converted to a date
IsNumeric(v)    True when v is a number or a string that can be converted to a number.
```

Common component properties

```
Rows: tblA  For each row in tblA, create a component.
Rows: txtB  For each txtB component, create one of me.
            I share txtB's data row and index.

Name        The name of the template, e.g. txtPerson.
txtPerson Ignore  Don't show me for the time being.
Form        My Form component. Read-only.
Main        The component that has my row. Read-only.
Index       0, 1, 2 … My index in my bundle. Read-only.
Last        The last index in my bundle.

Top         Pixels from my canvas top to my top.
Bottom      Pixels from my canvas top to my bottom.
Height      Pixels between my top and bottom border.

Left        Pixels from my canvas left to my left.
Right       Pixels from my canvas left to my right.
Width       Pixels between my left and right border.

BackColor   Color of my inner area.
BorderColor Color of my border.
Weight      Number of pixels across my border.

Visible     False if I am not visible. Default: True.

Canvas      The component where I am located. It may scroll and clip me. My left and top don't change when it scrolls. Default: My form.
ZOrder      Integer. On my canvas, I am above components with a lower ZOrder.
```

Common event handlers Act when the event occurs

Click, DoubleClick, KeyDown, KeyUp, KeyPress
MouseDown, MouseUp, GotFocus, LostFocus
MouseEnter, MouseMove, MouseLeave