Willkommen zum Informix Newsletter

Inhaltsverzeichnis

Aktuelles....1
TechTipp: Docker Images Developer Edition und Innovator-C....2
TechTipp: Informix Treiber für Python....2
TechTipp: Informix Abfrage mit Python....3
TechTipp: DATABAND – Überwachung von Daten Pipelines....4
TechTipp: Informix Auswertung „Sonne“ mit DATABAND Überwachung....5
TechTipp: SQLIDEBUG – Debugging Client-Server-Kommunikation....10
TechTipp: SQLIPRINT – Auswertung der Debug Informationen....11
TechTipp: Maximale Anzahl Datapages je Fragment....13
Gastbeitrag: Database Explorer für Informix - alte Pfade neu entdeckt....14
TechTipp: Version 14.10.FC10 ist verfügbar....23
Nutzung des INFORMIX Newsletters....23
Die Autoren dieser Ausgabe....24

Aktuelles

Liebe Leserinnen und Leser,

der Frühling zeigt sich aktuell von seiner besten Seite. Überall blüht es, und die Natur lebt auf. Auch die Redaktion des Informix Newsletters ist aus ihrer warmen Stube gekommen und hat sich umgeschaut, was es Neues und Interessantes gibt.
Dabei haben wir über den Tellerrand geschaut und stellen die Integration von Informix in eine Python Anwendung vor. Gut zum Thema passt das neue IBM Produkt Databand, das es ermöglicht, Datenströme und Pipelines zu überwachen.
Zudem freuen wir uns, dass wir einen Gastbeitrag erhalten haben, der einen komfortablen Weg zeigt, wie die Überwachung und Auswertung einer Informix Instanz mittels übersichtlicher Graphiken möglich ist.

Ihr TechTeam

TechTipp: Docker Images Developer Edition und Innovator-C

Lange Zeit fand sich die aktuelle Version der Informix Developer Edition und der Innovator-C Edition im DockerHub.
IBM hat dieses Repository verlassen und stellt ab sofort diese Editionen in einer eigenen IBM Container Registry (ICR) zur Verfügung.

Die Anforderung der aktuellen Versionen dieser Editionen kann damit über folgende Befehle erfolgen:

docker pull icr.io/informix/informix-innovator-c
(bisher: docker.io/ibmcom/informix-innovator-c)

docker pull icr.io/informix/informix-developer-database
(bisher: docker.io/ibmcom/informix-developer-database)

TechTipp: Informix Treiber für Python

Die Einbindung von Informix in unterschiedliche Entwicklungsumgebungen setzt als Basis die Installation der Software Informix C-SDK oder Informix JDBC voraus.

Für die Anbindung von Python an Informix sind zusätzlich zum C-SDK noch Treiber notwendig. Im aktuellen Beispiel nutzen wir den Treiber IfxPy, der unter github zur Verfügung steht:

https://github.com/OpenInformix/IfxPy.git

Für die Anbindung an Python ist das Informix C-SDK in einer aktuellen Version (ab 4.10) notwendig.

Das eigentliche Informix Modul für Python wird mittels

pip3 install ifxpy

installiert, und stellt die notwendigen Bibliotheken im Verzeichnis „site-packages“ des Benutzers bereit.
Im Readme der Seite steht, dass die aktuelle Version des Treibers die notwendigen Komponenten des Informix-C-SDK mit installiert. Dies war jedoch bei allen unseren Tests nicht der Fall, und es musste immer der Pfad zum bereits installierten Client-SDK als $INFORMIXDIR angegeben sein.

Alternativ steht der Treiber **pyodbc** zur Verfügung.

TechTipp: Informix Abfrage mit Python

Nachdem der Treiber für die Anbindung von Informix an Python installiert ist, sind Zugriffe auf die Informix Datenbank möglich.
Die für den Zugriff auf Informix notwendigen Zeilen sind „**fett**“ markiert:
(Beispiel: Abfrage Solarproduktion und Konsum)

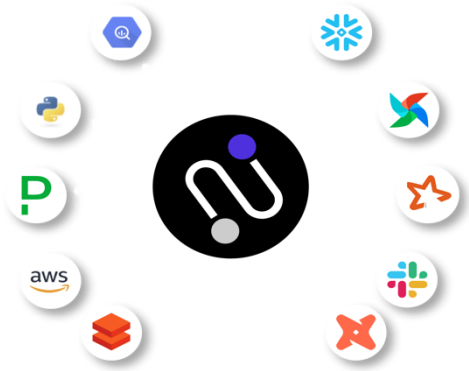
```
# Import pandas libraries
import pandas as pd
import csv

# Der INFORMIX Treiber wird mittels „import“ eingefügt
import IfxPyDbi as ifx


def read_sonne_informix():
        # Der Connection String enthält die Informationen für die Verbindung
        ConStr =
"SERVER=kalu;DATABASE=sonne;HOST=127.0.1.1;SERVICE=9088;UID=informix;PWD=passw0rd;DB_LOCALE=en_us.utf8;CLIENT_LOCALE=en_us.UTF8;"

        # Die Verbindung zur Informix Datenbank wird über ifx.connect aufgebaut
        conn = ifx.connect( ConStr, "", "")

        # Definition des Cursors zur SQL Abfrage
        cur = conn.cursor()
        cur.execute ("SELECT year(timestamp)||'_'                \
            ||lpad(month(timestamp),2,'0') as month              \
                sum(production)/1000 as production_kWh,          \
                sum(consumption)/1000 as consumption_kWh         \
                FROM sonne                                       \
            where timestamp >= '2022-08-01 00'                   \
                group by 1                                       \
                order by 1 "
        )

        # Mit der Funktion fetchall des Cursors werden die Daten übertragen
        sonnendata = cur.fetchall()
        return sonnendata


def prepare_sonnendata():
        # Call the step job - read data
        raw_data = read_sonne_informix()
        print(raw_data)

# Invoke the main function
prepare_sonnendata()
```

Das Ergebnis ist ein Array der Rückgabewerte:

```
[('2022_0811', Decimal('12.715'), Decimal('5.082')), ('2022_0812', Decimal('29.229'),
Decimal('9.378')), ('2022_0813', Decimal('29.75'), Decimal('9.92')), ('2022_0814',
Decimal('27.598'), Decimal('7.033')), ('2022_0815', Decimal('24.522'), Decimal('6.72')),
('2022_0816', Decimal('30.603'), Decimal('10.487')), ('2022_0817', Decimal('23.575'),
Decimal('8.554')),…
```

TechTipp: DATABAND – Überwachung von Daten Pipelines

Unter dem Motto „Über den Tellerrand geschaut“ betrachten wir heute einmal das Produkt DATABAND, das die Möglichkeit bietet „Data-In-Motion“ zu überwachen und ggf. bei abweichendem Verhalten einen Alarm auszulösen.
Dabei wird nicht nur überwacht, ob ein Job erfolgreich war, oder auf einen Fehler aufgelaufen ist. Die Überwachung kann zusätzlich die Laufzeit, die Datenmenge, sowie die Datenqualität umfassen und auf Abweichungen von den durchschnittlichen Werten reagieren.



DATABAND wird als Docker Image installiert und ist daher unkompliziert in jedes System integrierbar.

Aktuell werden folgende Applikationen direkt unterstützt:

- Python Pipelines (Einbinden von Databand Modulen)
- PySpark Pipelines (Einbinden von Databand Modulen)
- JVM (Set von Java Libraries zur Einbindung)
- Scale/Java Applications (Set von Java Libraries zur Einbindung)
- Airflow Cluster (SDK Databand Syncer)
- DataStage Next Generation (SDK Databand Syncer)

weitere Adapter erweitern mit jedem Release die unterstützten Applikationen.
Zudem kann mittels „custom integration“ eine eigene Anbindung erstellt werden.

Die Alarme werden in einer Übersicht gelistet und können zudem über Mail, Slack, Teams und andere Kanäle informieren.

DATABAND erkennt Zusammenhänge zwischen Applikationen und zeigt im Fehlerfall auf, in welchem Teil der Jobkette ein Fehler aufgetreten ist. Zudem wird über die Ergbnisse der Jobs der Zusammenhang hergestellt, der als „DataLineage“ erkennt, dass ein Folgejob eventuell auf nicht zuverlässigen Ergebnisse aufsetzt, falls eine der Datenquellen von einem vorhergehenden Job mit fehlerhaftem Ausgang erstellt wurde.

TechTipp: Informix Auswertung „Sonne“ mit DATABAND Überwachung

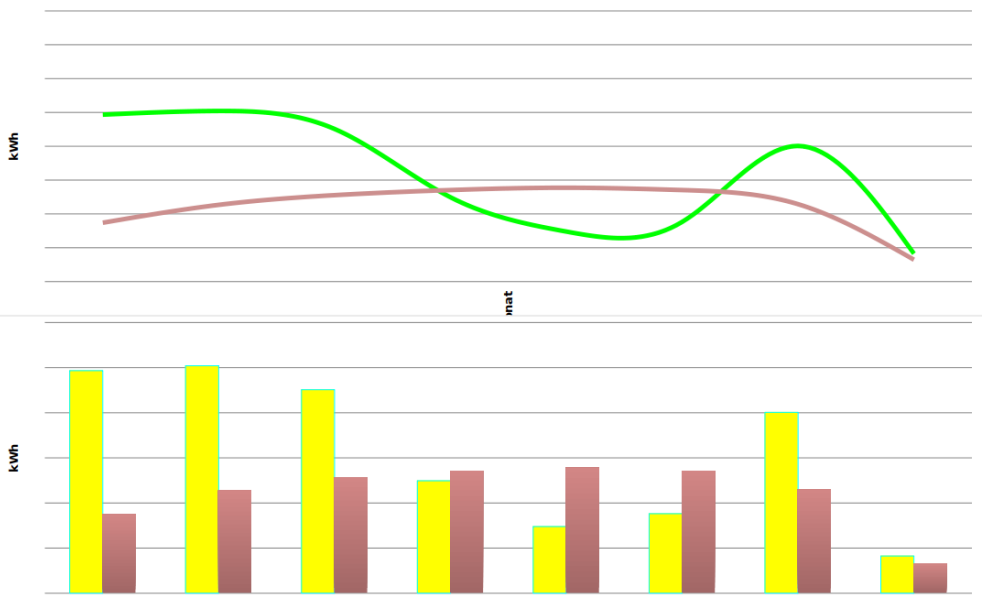
Das folgende Beispiel zeigt eine Datenabfrage mittels Python, in das die Überwachung der Abläufe mittels DATABAND integriert wurden. Das Script dient zur Analyse der Produktion einer Solaranlage und erstellt hierzu eine Graphik.

```
# Import pandas and databand libraries
import pandas as pd
import csv
import IfxPyDbi as ifx
import re
import openpyxl
from openpyxl.chart import BarChart, PieChart, LineChart, ScatterChart, Reference

# Databand wird über folgenden Import eingebunden:
from dbnd import dbnd_tracking, task, dataset_op_logger, log_duration, log_metric

# Um die Verbindung zu Databand einzurichten, wird die URL und ein User-Token verwendet:
databand_url = 'http://localhost:8080'
databand_access_token =
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTY3NjM3NzMzMywianRpIjoiMDQzNTk4Nj
ctOTI4My00ODlmLTgzYjctYWFjZmYyZDA1ZjhjIiwidHlwZSI6ImFjY2VzcyIsImlkZW50aXR5Ijoia2FsdSIsIm5iZiI6MTY3Nj
M3NzMzMywiZXhwIjoxNzM5NDQ5MzMyLCJ1c2VyX2NsYWltcyI6eyJlbnYi42IifX0.sisLSBDTiAg3a2fe7ic41_7XKSCzQx9T6w
XwdtZxR4E'


# mittels @task wird ein Task für das Logging markiert
@task
def read_sonne_informix():
    # Unique name for logging
    unique_file_name = FILE_NAME
    # Log the data read
    metric_name = 'Sonnenwerte_read'
    # Hier wird die Zeit für das Lesen der Daten erfasst:
    with log_duration('Read SUN Data from INFORMIX') as logger:
        ConStr =
"SERVER=kalu;DATABASE=sonne;HOST=127.0.1.1;SERVICE=9088;UID=informix;PWD=passw0rd;DB_LOCALE=en_us.ut
f8;CLIENT_LOCALE=en_us.UTF8;"
        conn = ifx.connect( ConStr, "", "")

        cur = conn.cursor()
        cur.execute ("SELECT year(timestamp)||'_'          \
                  ||lpad(month(timestamp),2,'0') as month  \
                 sum(production)/1000 as production_kWh,   \
                 sum(consumption)/1000 as consumption_kWh  \
                 FROM sonne                                \
                 where timestamp >= '2022-08-01 00'        \
                 group by 1                                \
                 order by 1 "
        )
        sonnendata = cur.fetchall()
        # Hier wird die Metric für die Anzahl der Records erstellt:
        log_metric(metric_name,len(sonnendata))
    return sonnendata

# in einem weiteren Task schreiben wir die Daten in eine CSV Datei:
@task
def write_sonne_csv(raw_data):
    unique_file_name_1 = 'Sonnen_Daten.csv'
    # Log writing the csv
    with dataset_op_logger(unique_file_name_1,"write",with_schema=True,with_preview=True) as logger:
        # Write the csv file
        fields = ['Datum','Konsum']
        with open(unique_file_name_1, 'w') as f:
           write = csv.writer(f)
           write.writerow(fields)
           write.writerows(raw_data)
        logger.set(data=f)
```

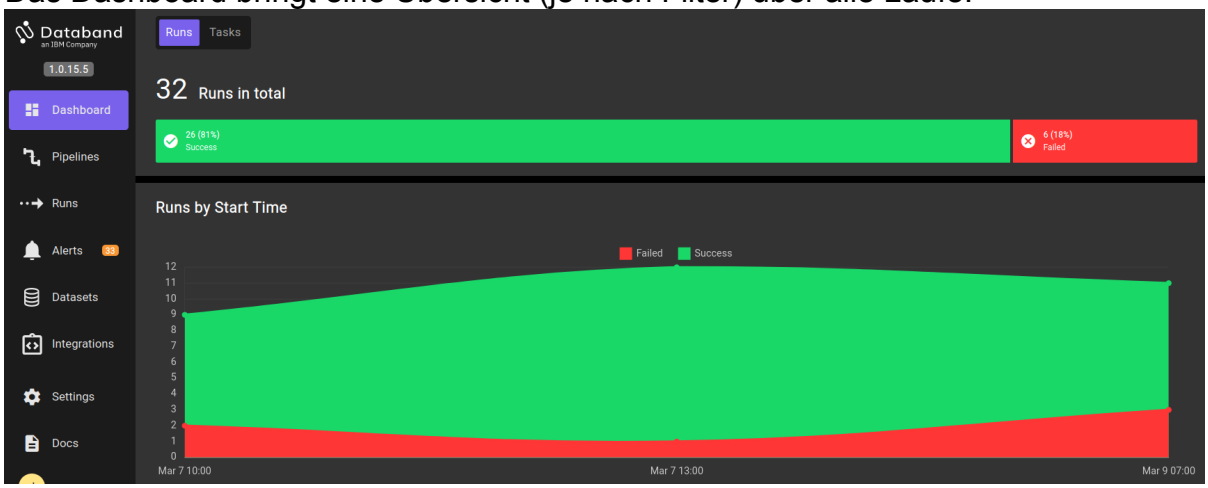
```
# in einem weiteren Job erstellen wir eine Chart aus den Daten:
@task
def create_sonnen_chart(raw_data):
        file='sonnen_produktion.xlsx'
        wb=openpyxl.Workbook()
        sheet=wb.active
        x=1
        for row in raw_data:
                new_cell1=sheet.cell(x,1)
                new_cell1.value=row[0]
                new_cell2=sheet.cell(x,2)
                new_cell2.value=row[1]
                new_cell3=sheet.cell(x,3)
                new_cell3.value=row[2]
                x+=1
        values=Reference(sheet,
                min_col=2,
                min_row=1,
                max_col=3,
                max_row=sheet.max_row
        )
        print(values)
        chart=LineChart()
        chart.add_data(values)
        chart.x_axis.title='Monat'
        chart.y_axis.title='kWh'
        chart.y_axis.scaling.min=0
        chart.y_axis.scaling.max=50
        chart.style=20
        chart.size=800
        chart.width=30
        chart.height=10
        chart.legend=None

        s=chart.series[0]
        s.graphicalProperties.line.solidFill = "00ff00"
        s.graphicalProperties.solidFill = "ff0000"

        sheet.add_chart(chart, 'D1')
            wb.save(file)


def prepare_sonnendata():
    with dbnd_tracking(
             conf={
                 "core": {
                     "databand_url": databand_url,
                     "databand_access_token": databand_access_token,
                 }
             },
             job_name="sonnen_data" + unique_suffix,
             run_name="daily",
             project_name="Sonne Analytics" + unique_suffix,
    ):
        # Call the step job - read data
        raw_data = read_sonne_informix()

        # Write data by product line
        write_sonne_csv(raw_data)

        # Chart creation
        create_sonnen_chart(raw_data)

        print("Finished running the pipeline")

# Invoke the main function
prepare_sonnendata()
```

Als Ergebnis erstellen wir neben einer CSV-Datei der Werte auch eine Graphik.
(Werte im Beispiel mit Stand 08.03.2023)

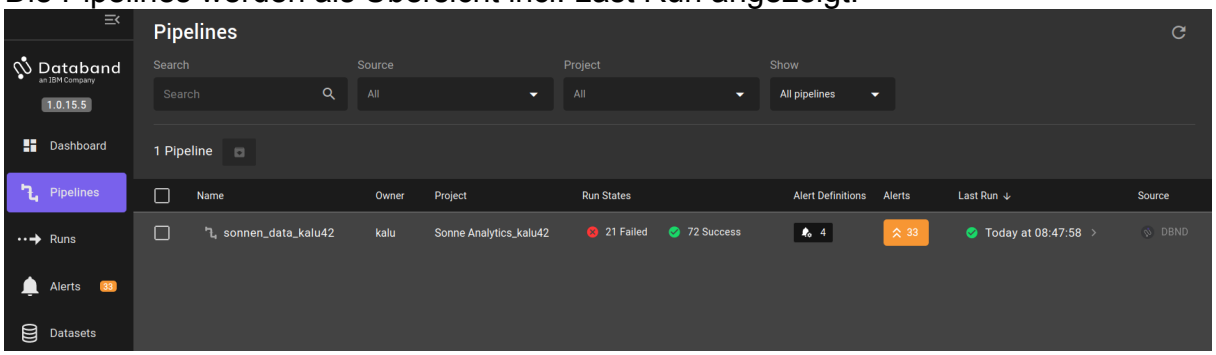
2022_08	493,434	173,963
2022_09	504,243	226,739
2022_10	451,149	255,414
2022_11	249,302	270,522
2022_12	147,793	277,358
2023_01	176,383	270,501
2023_02	400,947	228,808
2023_03	82,427	64,62



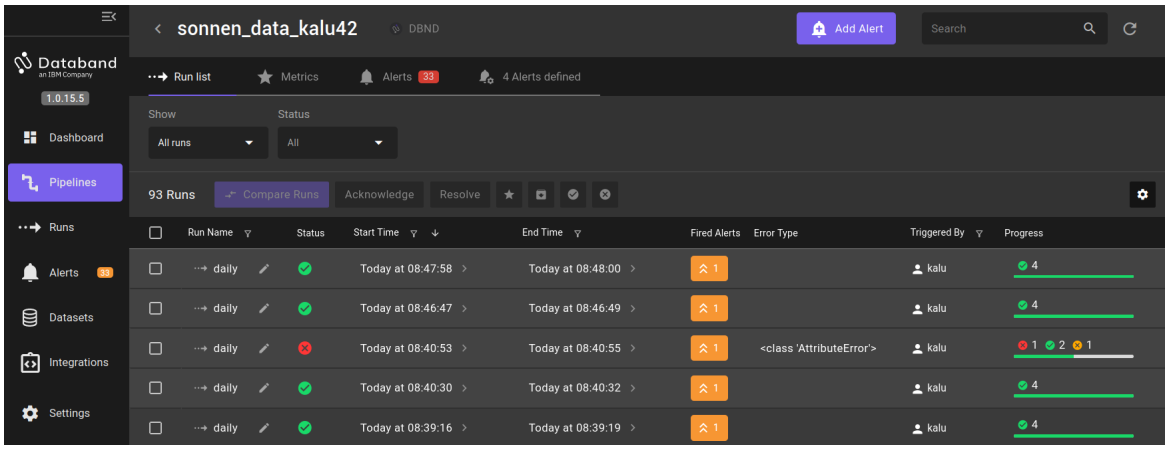
Im Monitoring von DATABAND sind die Aufrufe im Detail zu sehen:
Das Dashboard bringt eine Übersicht (je nach Filter) über alle Läufe.



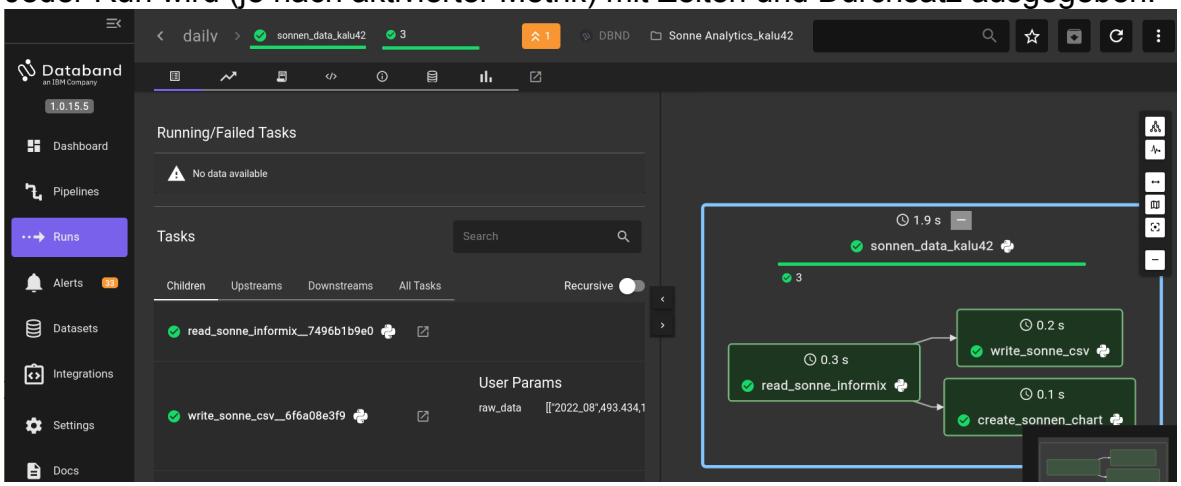
Die Pipelines werden als Übersicht incl. Last Run angezeigt:



Details zu den einzelnen Läufen:



Jeder Run wird (je nach aktivierter Metrik) mit Zeiten und Durchsatz ausgegeben:

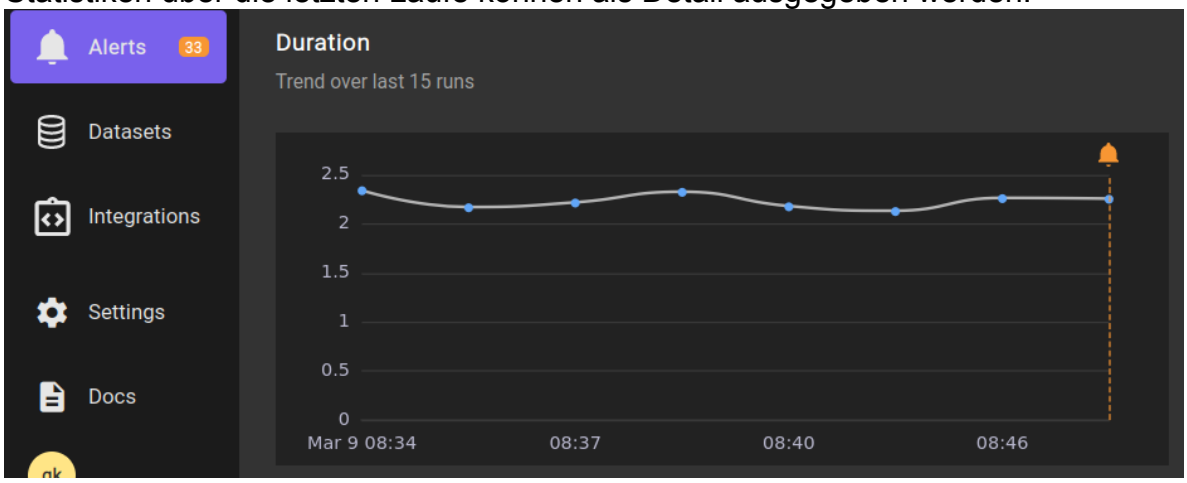


Im Beispiel wurde ein Alarm gesetzt, der bei einer Gesamtzeit von mehr 2 Sekunden zuschlägt (Laufzeit hier: 2,25sec). Der Alarm wurde auch an Slack übertragen.

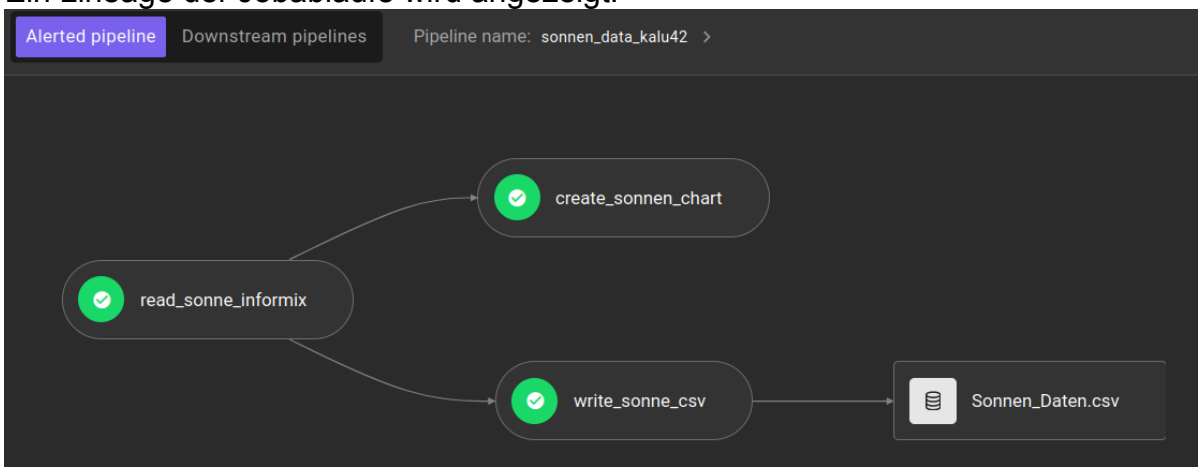
Alarme können mit unterschiedlicher Priorität erstellt werden.
Hierbei sind als Optionen sowohl Abweichungen von vorgegebenen Grenzen, als auch Abweichungen in % vom Durchschnitt der erfolgreichen Läufe möglich.

Fehlerhafte Läufe werden immer als Alarm gemeldet.

Statistiken über die letzten Läufe können als Detail ausgegeben werden:



Ein Lineage der Jobabläufe wird angezeigt.



Bei Fehlern innerhalb der Jobkette wird der verursachende Job markiert und es wird die Fehlermeldung (so weit möglich) mit ausgegeben, was die Fehlersuche sehr vereinfacht.

TechTipp: SQLIDEBUG – Debugging Client-Server-Kommunikation

In der Kommunikation zwischen Client und Server wird eine Reihe an Informationen ausgetauscht. Ist diese Kommunikation gestört, oder weist eine unerwartet schlechte Performance auf, so besteht die Möglichkeit diese Kommunikation zu protokollieren. Hierfür wird die Umgebungsvariable SQLIDEBUG in der Umgebung des Clients gesetzt.

```
export SQLIDEBUG=2:/tmp/sqlidebug.out
```

Der erste Wert gibt das Format an:

1: Es wird ein Hexdump der Kommunikation erstellt
2: Es wird ein binäres Format erstellt, das mittels sqliprint eine lesbare ASCII-Ausgabe erzeugen kann.

Der zweite Wert benennt die Datei, in der die Ausgabe gespeichert werden soll. Diese wird mit einer eindeutigen ID erweitert.

Die Datei, die mit der Option 1 erstellt wird, hat im Beispiel folgenden Inhalt:

```
 S->C (0x2185fa0) 4
   0  6e   1  3c
 C->S (0x2185fa0) 16
   0  7e   0   9  ff  fe  df  fe  74  aa  62  13  c0   0   0   c

 S->C (0x2185fa0) 16
   0  7e   0   9  bd  be  9f  fe  7f  b7  ff  ef  f8   0   0   c

 C->S (0x2185fa0) 240
   0  51   0   6   0  e6   0   c   0  74   0   6  44  42  4c  41
  4e  47   0   a  65  6e  5f  75  73  2e  75  74  66  38   0   6
  44  42  54  45  4d  50   0   4  2f  74  6d  70   0   5  53  48
  45  4c  4c   0   0   9  2f  62  69  6e  2f  62  61  73  68   0
   0   b  53  55  42  51  43  41  43  48  45  53  5a   0   0   2
  31  30   0   6  44  42  50  41  54  48   0   1  2e   0   0   4
  50  41  54  48   0  74  2f  6f  70  74  2f  69  6e  66  6f  72
  6d  69  78  2f  62  69  6e  3a  2f  75  73  72  2f  6c  6f  63
  61  6c  2f  73  62  69  6e  3a  2f  75  73  72  2f  6c  6f  63
  61  6c  2f  62  69  6e  3a  2f  75  73  72  2f  73  62  69  6e
  3a  2f  75  73  72  2f  62  69  6e  3a  2f  73  62  69  6e  3a
  2f  62  69  6e  3a  2f  75  73  72  2f  67  61  6d  65  73  3a
  2f  75  73  72  2f  6c  6f  63  61  6c  2f  67  61  6d  65  73
  3a  2f  73  6e  61  70  2f  62  69  6e   0   8  4e  4f  44  45
  46  44  41  43   0   3  79  65  73   0   0   0   0   0   0   c

 S->C (0x2185fa0) 2
   0   c
…
```

TechTipp: SQLIPRINT – Auswertung der Debug Informationen

Das Script sqliprint wird mit dem Informix Client-SDK ausgeliefert.
Hiermit kann die Protokollierung einer Client-Server-Verbindung in ASCII-Text umgewandelt werden.
Enthalten sind sowohl die SQL-Befehle, als auch die Rückgabeinformationen.
Jeder Teilschritt wird mit einer Zeitmarke versehen.

Die Aufbereitung der gesammelten Informationen erfolgt mittels:

```
sqliprint /tmp/sqlidebug.out.4211 > /tmp/sqliprint.out
```

Die somit erstellte Datei hat im Beispiel folgenden Inhalt:

```
SQLIDBG Version 1

S->C (4)                                Time: 2023-03-10 09:54:04.97264
        SQ_INTERNALVER
                Internal Version Number: 316

C->S (16)                               Time: 2023-03-10 09:54:04.97270
        SQ_PROTOCOLS
                 Cprotocol stream len = 9
                 byte[0] = ff
                 byte[1] = fe
                 byte[2] = df
                 byte[3] = fe
                 byte[4] = 74
                 byte[5] = aa
                 byte[6] = 62
                 byte[7] = 13
                 byte[8] = c0
        SQ_EOT

S->C (16)                               Time: 2023-03-10 09:54:04.97288
        SQ_PROTOCOLS
                 Sprotocol stream len = 9
                 byte[0] = bd
                 byte[1] = be
                 byte[2] = 9f
                 byte[3] = fe
                 byte[4] = 7f
                 byte[5] = b7
                 byte[6] = ff
                 byte[7] = ef
                 byte[8] = f8
        SQ_EOT

C->S (240)                              Time: 2023-03-10 09:54:04.97298
        SQ_INFO
                INFO_ENV
                        Name Length = 12
                        Value Length = 116
                        "DBLANG"="en_us.utf8"
                        "DBTEMP"="/tmp"
                        "SHELL"="/bin/bash"
                        "SUBQCACHESZ"="10"
                        "DBPATH"="."
                        "PATH"="/opt/informix/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
                        "NODEFDAC"="yes"
                INFO_DONE
        SQ_EOT

…
S->C (28)                               Time: 2023-03-10 09:54:04.97405
        SQ_DONE
                Warning..: 0x15
```

```
                # rows...: 0
                rowid....: 0
                serial id: 0
        SQ_COST
                estimated #rows: 1
                estimated I/O..: 1
        SQ_EOT

C->S (46)                               Time: 2023-03-10 09:54:04.97420
        SQ_PREPARE
                # values: 0
                CMD.....: "select count(*) from systables
" [31]
        SQ_NDESCRIBE
        SQ_WANTDONE
        SQ_EOT

S->C (72)                               Time: 2023-03-10 09:54:04.97457
        SQ_DESCRIBE
                Stmt Type...........: 2
                Server Stmt Id......: 0
                Estimated Cost......: 0
                Size of output tuple: 9
                # output fields.....: 1
                Size of string table: 11
                0) Field '(count(*))'
                        Index into string table: 0
                        Starting offset in tuple: 0
                        Type.....................: DECIMAL; NOT NULLABLE
                        Length : 3840 (0xf00)
        SQ_DONE
                Warning..: 0x0
                # rows...: 0
                rowid....: 0
                serial id: 0
        SQ_COST
                estimated #rows: 1
                estimated I/O..: 1
        SQ_EOT

C->S (20)                               Time: 2023-03-10 09:54:04.97463
```

TechTipp: Maximale Anzahl Datapages je Fragment

Informix lässt sich nahezu unbegrenzt skalieren, und die Grenze für die Gesamtgrösse der DBSpaces liegt bei 131 PetaByte. Die maximale Anzahl der Datenbanken beträgt 21 Millionen je Instanz, und die maximale Anzahl Tabellen ist 477 102 080.
Diese Werte stammen aus der Dokumentation der Version 14.10.

Diese Grenzen werden üblicherweise nicht zum Problem.
Das Limit von 16 775 134 DataPages je Fragment hingegen trifft immer wieder Kunden, die grosse Tabellen nutzen, in denen auch viele Altdaten gehalten werden (müssen).

Für diese Herausforderung gibt es unterschiedliche Lösungen.
Die einfachste Lösung ist es, die Tabelle zu fragmentieren bzw. zu partitionieren und damit dieses Limit auf ein Vielfaches zu erhöhen. Dies kann mit bestehenden Tabellen über den Befehl „alter fragment on table …“ erfolgen.

Die Fragmentierung ist jedoch (lizenztechnisch) auf die Enterprise Edition beschränkt und darf in der Workgroup Edition nicht genutzt werden.

Für viele Fälle bietet sich eine andere Lösung an:
Statt der Default Pagesize von 2k (bzw. 4k, je nach Betriebssystem), können die Daten in DBSpaces mit einer Pagesize von 8k, 10k, 12k, 14k oder 16k gespeichert werden.
Damit passen bis zu 8 Mal mehr Daten in ein Fragment.
Um eine Tabelle in einen DBSpace mit einer anderen PageSize zu verschieben, kann der Befehl „alter fragment on table xxx init in <dbspaces16k>“ verwendet werden.
Dies bedingt jedoch, dass ausreichend Platz im Transaktionslogs zur Verfügung steht, um diesen Befehl und dessen Aktionen zu protokollieren.

Alternativ hierzu kann eine neue Tabelle im grösseren DBSpace erstellt werden. Diese wird für das Kopieren der Daten in den Mode „raw“ versetzt, um das Transaktionslog für das Kopieren zu deaktivieren. Sind alle Daten kopiert, so wird die Tabelle wieder in den Mode „standard“ geändert. Anschliessend kann die alte Tabelle als Sicherheitskopie umbenannt werden (z.B. auf <tabname>_old) und die neue Tabelle wird mittels „rename table“ auf den bisherigen Tabellennamen verschoben.
Abschliessend sind noch ggf. die Indizes und Constraints anzupassen, die mit der bisherigen Tabelle verknüpft waren, und nun in der neuen Tabelle eingepflegt werden müssen.

Gastbeitrag: Database Explorer für Informix - alte Pfade neu entdeckt

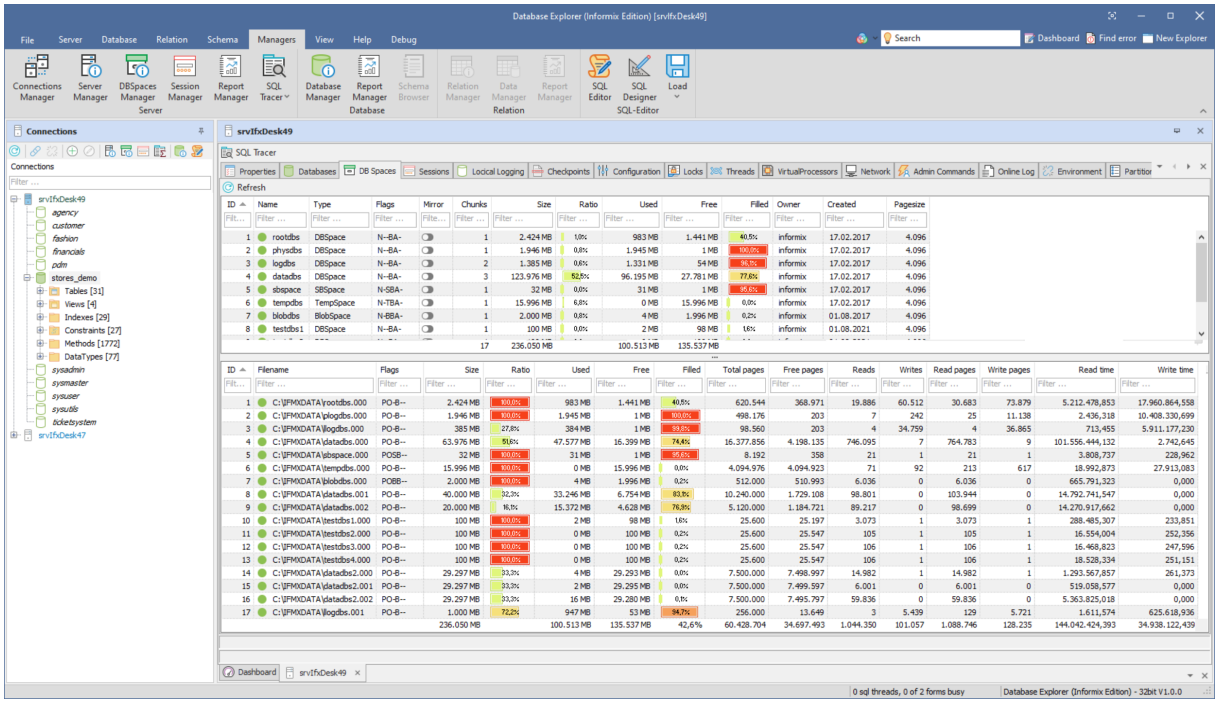
Anmerkung der Redaktion:

Herr Oliver Ickler der iBiSS Software Solutions GmbH hatte bereits vor einiger Zeit in einem IUG WebCast das Tool „Database Explorer“ vorgestellt. Das Interesse der Teilnehmer war sehr rege, so dass wir uns freuen, heute einen Gastbeitrag zum Thema veröffentlichen zu dürfen.
Inzwischen gibt es eine allgemein verfügbare, aktuelle Version.
Im folgenden Beitrag wird eine Auswahl der Möglichkeiten vorgestellt.
Die Kontaktdaten finden sich am Ende des Beitrags.

Informix ist eine besondere Datenbank. Doch Informix-Administratoren und Entwickler mussten sich bisher bei der Werkzeugwahl immer auf Kompromisse einlassen. Standarddatenbankwerkzeuge sprechen nicht vollständig Informix. Informix-spezifische Werkzeuge unterstützen nur einen Teil von dem, was man braucht. Und eine Datenbank über die Kommandozeile zu verwalten ist für das Jahr 2023 auch nicht mehr zeitgemäß. Doch nun gibt es endlich den **Database Explorer**.
Der Database Explorer ist ein auf Informix spezialisiertes All-in-One Datenbankwerkzeug. Diese native Windows-Anwendung bietet eine grafische Oberfläche für Serververwaltung, Datenbankadministration, SQL-Erstellung, Datenabfragen, Reports, SQL-Tracing und vieles mehr in einem modernen Look. Er ist das Ergebnis aus 25 Jahren Informix Erfahrung.

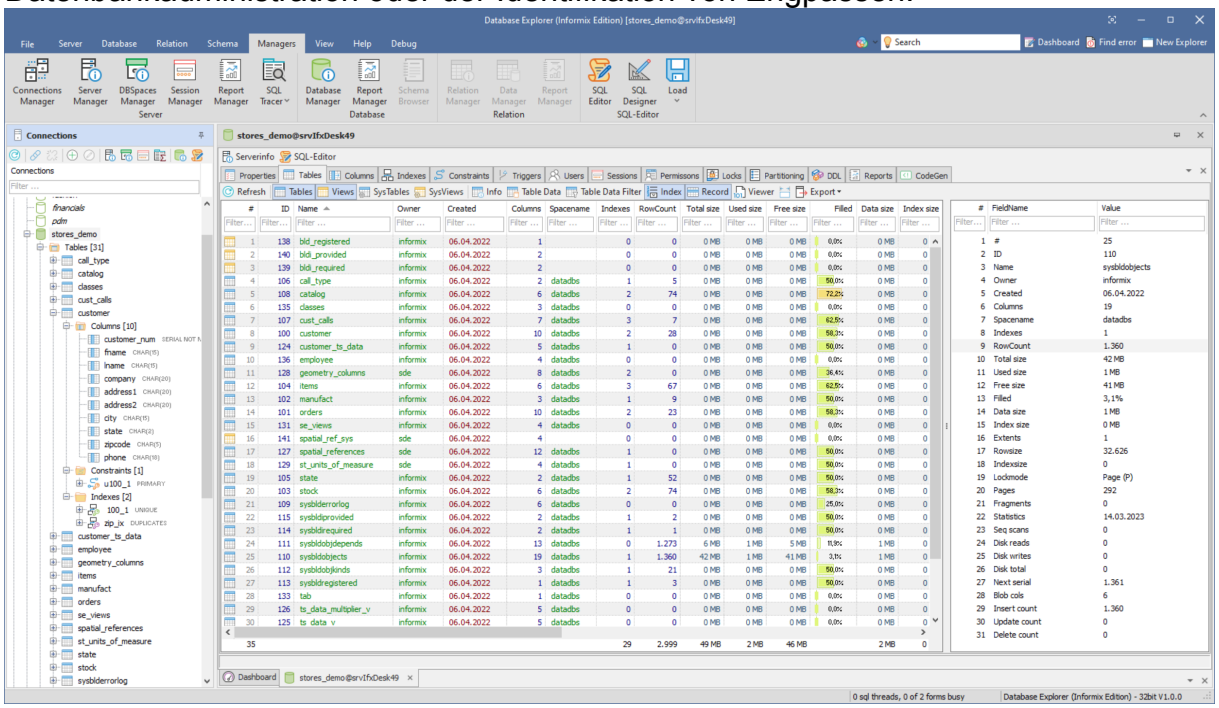
Servermanagement – alles auf einen Blick

Mit Hilfe der integrierten Manager findet der Anwender alles über seine Datenbanken: Speicherverbrauch, DB-Spaces, Logging, Locks, Threads, Online-Log, SQL Tracing sowie viele weitere datenbankspezifische und datenbankübergreifende Informationen.



Darstellung der DB-Spaces und Chunks im Servermanager

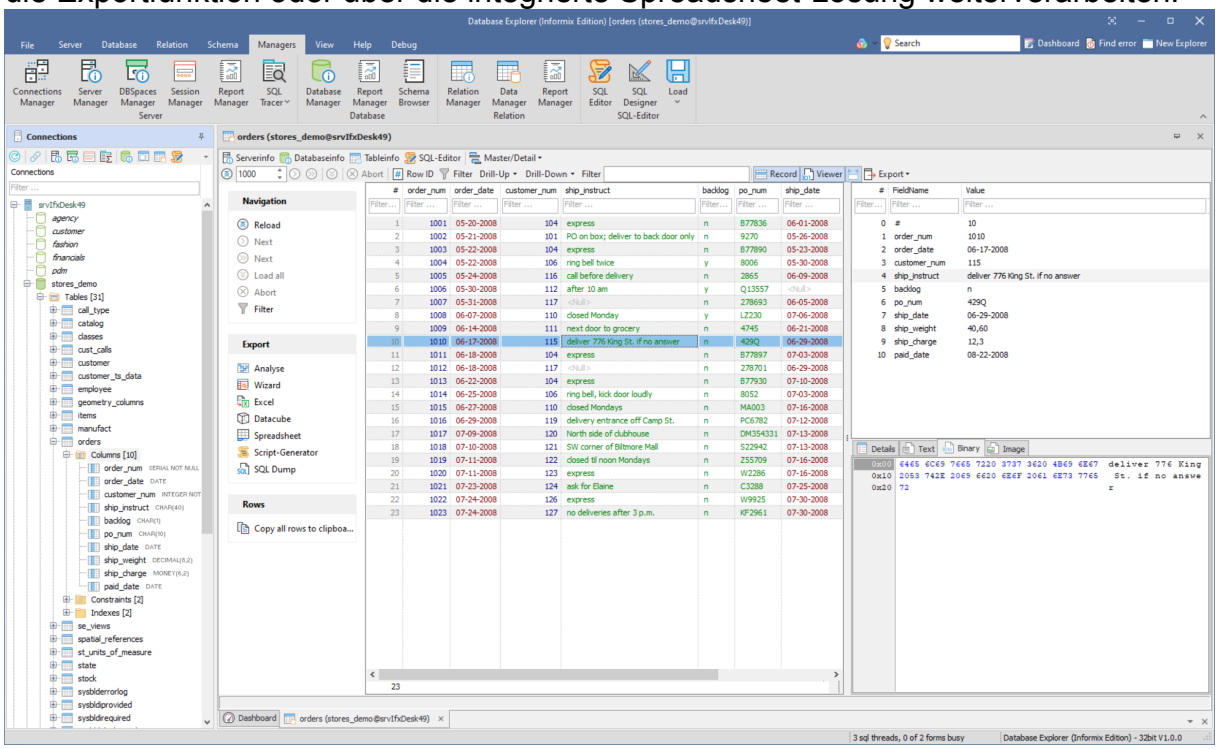
Mit einem Klick gelangt man von Server zur Datenbank, von der Datenbank zur Tabelle und zurück. Ein guter Überblick über alle Komponenten hilft bei der Datenbankadministration oder der Identifikation von Engpässen.



Darstellung aller Tabellen und deren Eigenschaften im Datenbankmanager

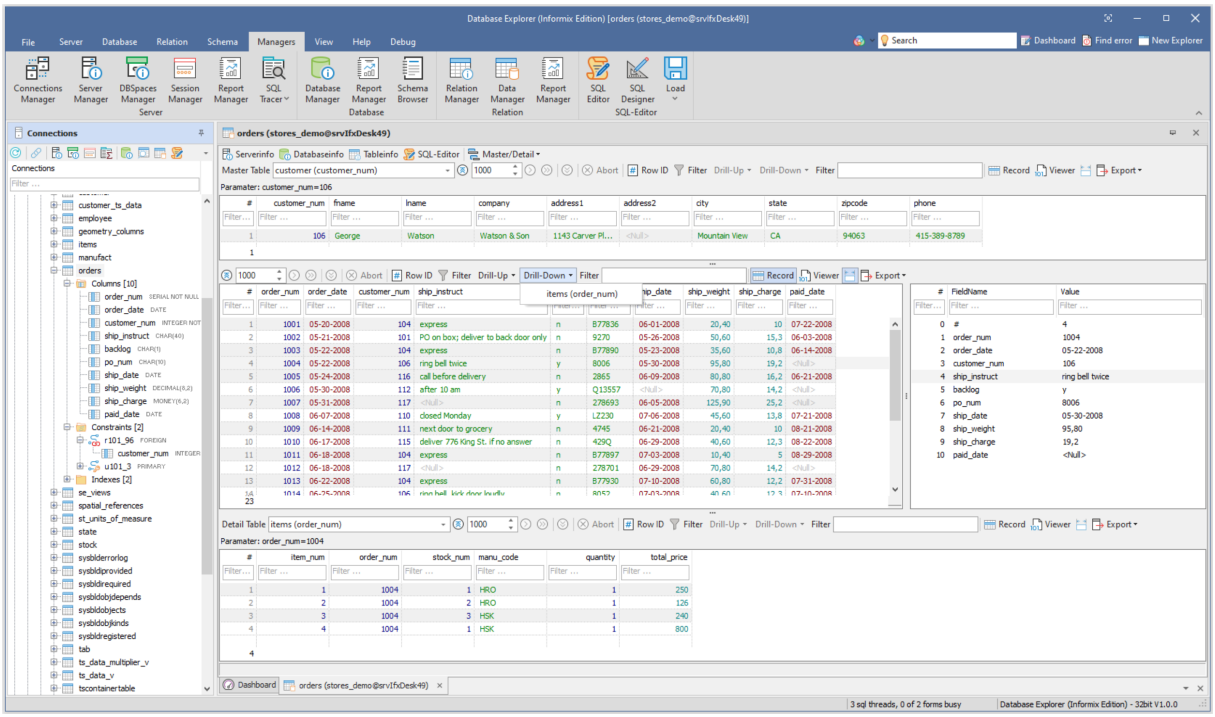
Datenabfragen – für professionelle Datenanalyse

Möchte man in die Daten abtauchen, bietet der Datenmanager eine grafisch übersichtliche Darstellung. Die farbliche Repräsentation der Daten kann in verschiedenen Formaten erfolgen. Zeilen oder einzelne Feldinhalte können in Detailanzeigen als Text, Binär oder Bild dargestellt werden. Mit Filtern, Suchen oder der integrierten Pivot-Funktion lassen sich Datenanalysen schnell und einfach erledigen. Ergebnisse lassen sich bequem über die Exportfunktion oder über die integrierte Spreadsheet-Lösung weiterverarbeiten.



Darstellung der Inhalte einer Tabelle

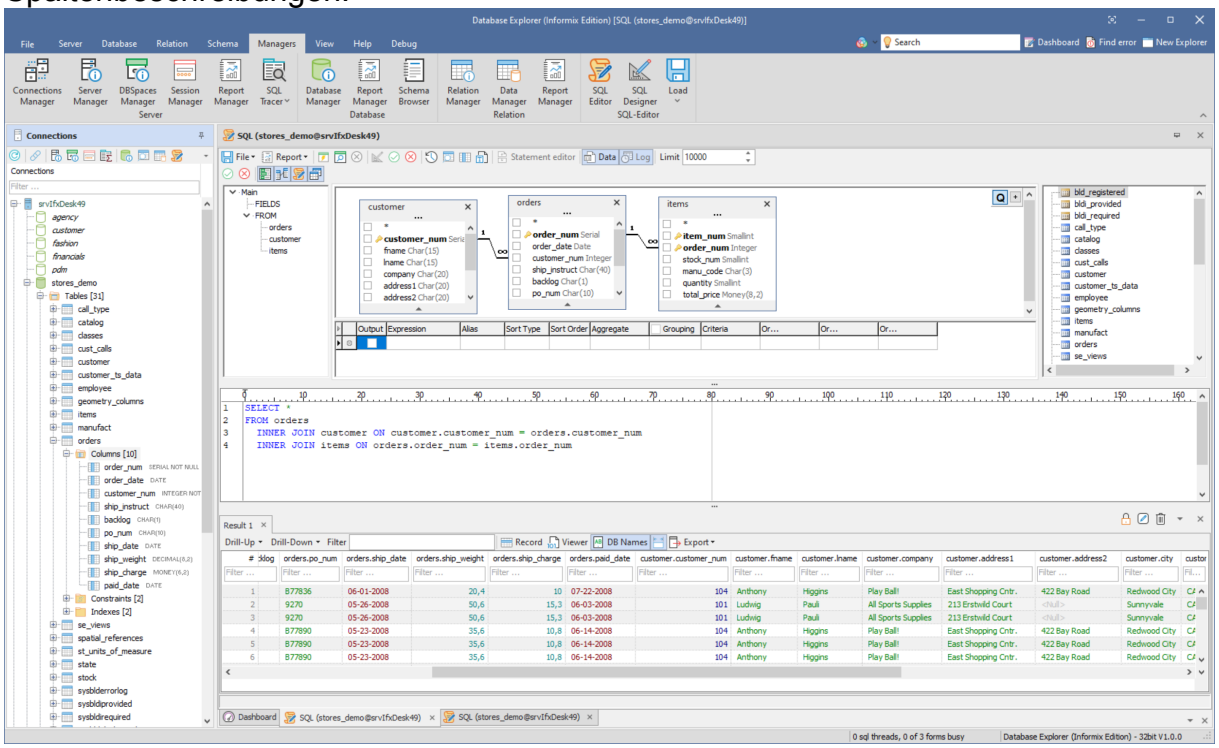
Eine Besonderheit ist die Drill-Up/Down Navigation, die über Fremdschlüssel mit einem Klick referenzierte Daten aus einer anderen Tabelle anzeigen kann. Zusätzlich kann der Anwender bis zu drei Ebenen verbundener Tabellen in einer Anzeige gleichzeitig betrachten.



Master / Detail Darstellung über bis zu drei Ebenen.

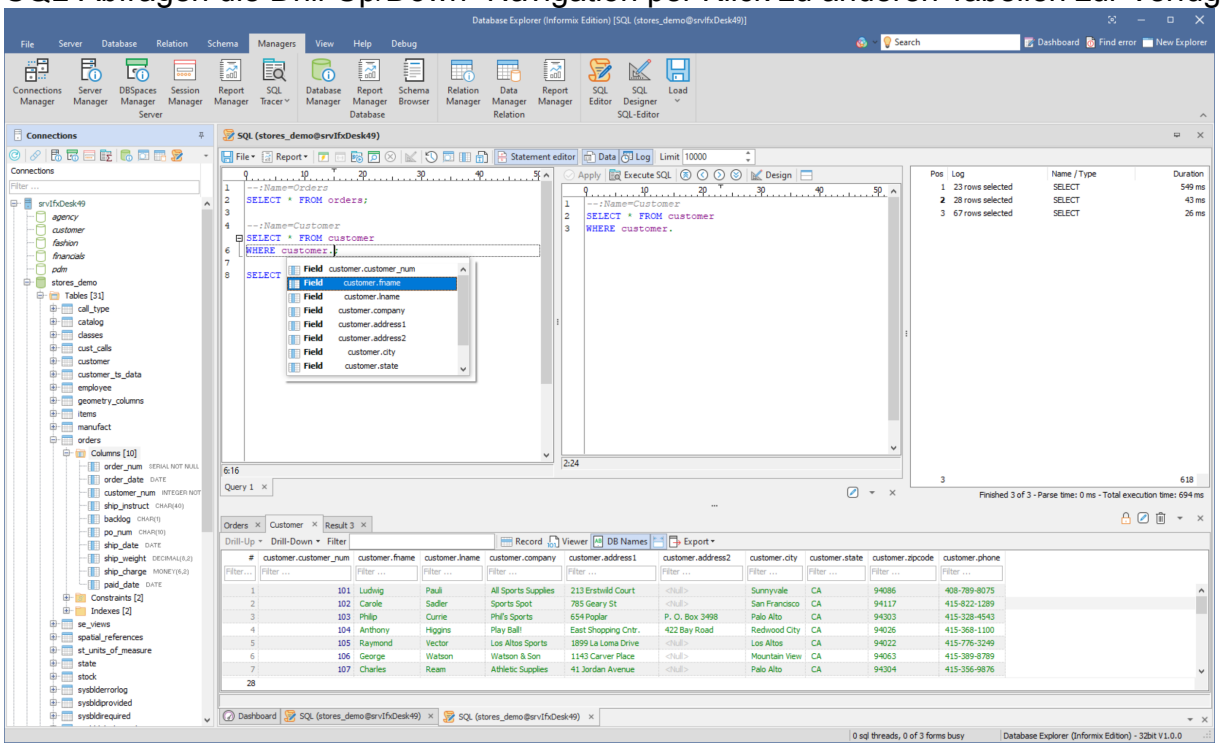
SQL-Erstellung – nie war es einfacher Abfragen zu erstellen

Der SQL-Editor bietet neben Syntax-Highlighting und Autovervollständigung auch einen visuellen Drag&Drop SQL-Designer. Mehrere Statements in einer SQL-Abfrage werden im Ergebnis in separaten Tabs dargestellt und unabhängig voneinander protokolliert. Weitere Features sind Templates, Exporte, Abfragehistorie und eigene Tabellen oder Spaltenbeschreibungen.



Visueller SQL-Designer

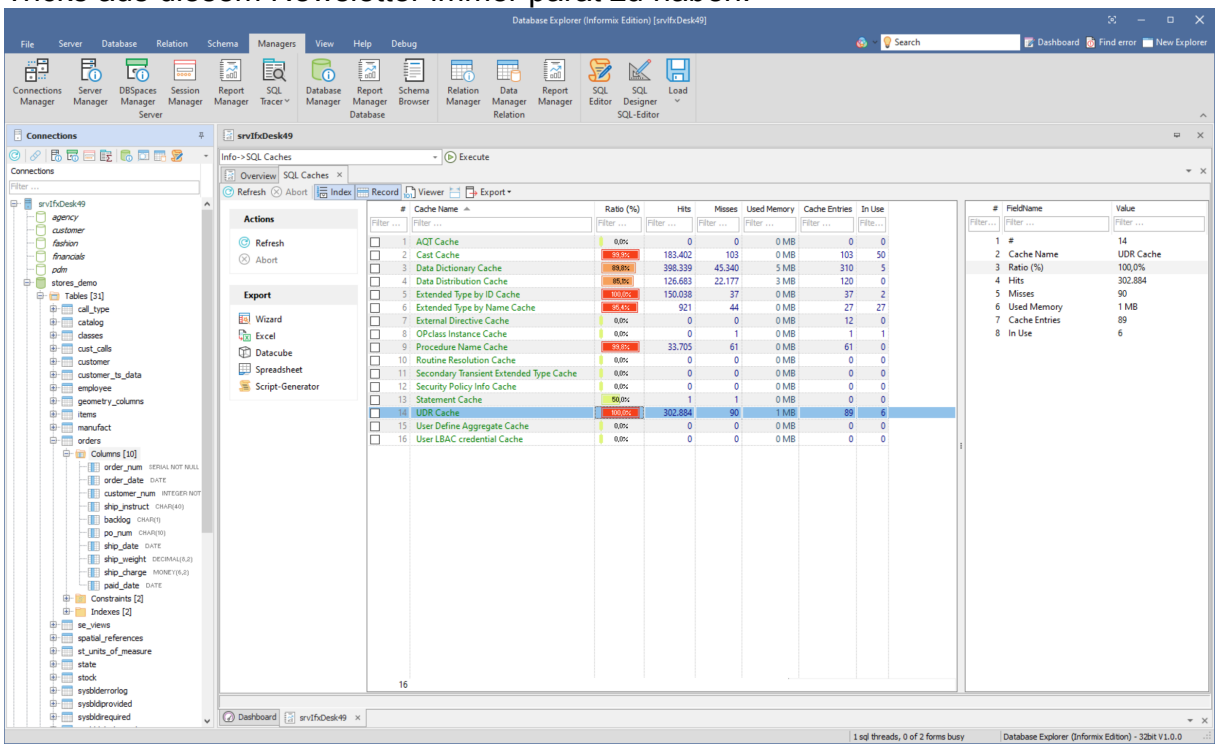
Der integrierte Query-Analyzer untersucht Abfragen und liefert über Schemainformationen dem Anwender Metainformationen über Spalten und Tabellen. Dadurch steht auch bei SQL-Abfragen die Drill-Up/Down- Navigation per Klick zu anderen Tabellen zur Verfügung.



SQL- und Statement-Editor

Reportmanager – für regelmäßige Routinen

Im Reportmanager stehen vordefinierte Auswertungen zur Ausführung bereit. Es können eigene Reports mit frei wählbaren Spalten, Parametern und Standardwerten erstellt werden. Fortgeschrittene Anwender können auch Reports per Pascal Script programmieren. Reports eignen sich auch hervorragend, um zukünftig die Tipps und Tricks aus diesem Newsletter immer parat zu haben!

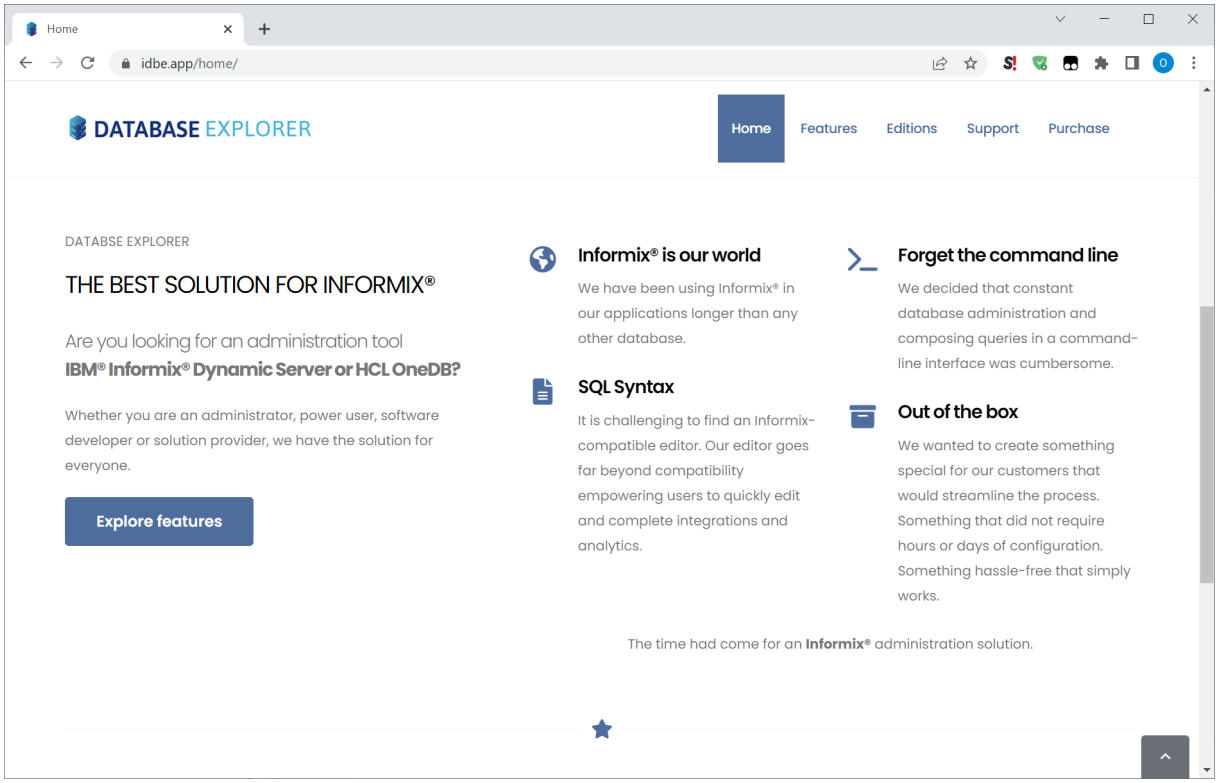


Report Manager mit tabellarischer Darstellung der Daten

Erleben Sie den Database Explorer selbst!
Sind Sie neugierig geworden? Der Database Explorer steht in drei Editionen zur Verfügung: Starten Sie mit der kostenlosen Community Edition oder probieren Sie die Professionell oder Enterprise Edition mit der vollfunktionsfähigen 30-Tage-Testversion aus.

Weitere Informationen und alle Details zu den unterschiedlichen Editionen finden Sie unter **www.idbe.app**.

Zum Produktstart erhalten alle Newsletter-Abonnenten einen Rabatt von 20% auf jede Bestellung per Rechnung! Schicken Sie eine E-Mail mit Angabe der Edition und dem Kennwort „INFORMIX NEWSLETTER“ bis zum 30.04.2023 an idbe@ibiss.de.



Für weitere Fragen zum Tool steht Oliver Ickler gerne zur Verfügung:

iBiSS Software Solutions GmbH
Oliver Ickler
Pulverstraße 16
42899 Remscheid

Telefon: +49(0)2191/589458-13
eMail: oickler@ibiss.de
Internet: https://www.ibiss.de

TechTipp: Version 14.10.FC10 ist verfügbar

Seit wenigen Tagen ist die Version 14.10.FC10 auf allen unterstützten Plattformen verfügbar.
Neben einigen Problembehebungen sind im Release folgende Änderungen implementiert:

- Interne IBM Java Version wurde auf JRE 8.0.11.20 erhöht (ausser auf HP-UX)
- Das GSKit enthält nun auch auf Debian/Ubuntu wieder die notwendigen Packages
- Das ClientSDK 4.50.xC10 wurde für Linux Intel 32 bit bereitgestellt.

Nutzung des INFORMIX Newsletters

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit.
Die IUG hat sich dankenswerterweise dazu bereit erklärt, den INFORMIX Newsletter auf ihren Web Seiten zu veröffentlichen.
Da uns weder Tippfehler noch Irrtümer fremd sind, bitten wir hier um Nachsicht falls sich bei der Recherche einmal etwas eingeschlichen hat, was nicht wie beschrieben funktioniert.
Rückmeldungen hierzu sind herzlich Willkommen !
Die gefundenen Tippfehler dürfen zudem behalten und nach Belieben weiterverwendet werden.
Eine Weiterverbreitung in eigenem Namen (mit Nennung der Quelle) oder eine Bereitstellung auf der eigenen HomePage ist ausdrücklich erlaubt. Alle hier veröffentlichen Scripts stehen uneingeschränkt zur weiteren Verwendung zur Verfügung.

Die Autoren dieser Ausgabe

Andreas Legner
INFORMIX Advanced Support
HCL Technologies

Martin Fuerderer
INFORMIX Development
HCL Technologies

Gerd Kaluzinski IBM Expert Lab DACH Consultant
Data & AI Software
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Gastbeitrag von **Oliver Ickler** der Firma iBiSS Software Solutions GmbH

Dank auch an die vielen Helfer im Hintergrund.

Nicht zu vergessen der Dank an die Informix User Group, ohne die es keinen Neuanfang des INFORMIX Newsletters gegeben hätte und die dankenswerter Weise die Verteilung übernimmt.

Foto Nachweis:
Blick vom Karren bei Dornbirn im März 2023 (Gerd Kaluzinski)