Db2, SQL, and Open source languages on IBM i

Stephanie Rabbani and Alan Seiden

Db2 Lessons from the Trenches of Open Source on IBM i

Database-centric Db2 techniques (thanks, Scott and Db2 team!)...

…that make open source and scripting languages easier in a programmer's world

Stephanie Rabbani

- CTO, Seiden Group
- PHP and Python developer
- Contributor to PHP toolkit, Python toolkit, other open source repos
- Long time user of DB2 and SQL (15+ years)

IBM CHAMPION

Alan Seiden

IBM CHAMPION

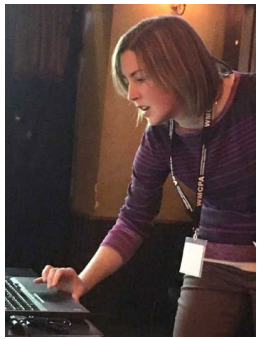
- Principal, Seiden Group
- Mentor to CIOs and development teams
- Delivers modern technical solutions
- Host and sponsor of CIO Summit (rsvp@seidengroup.com)
- Open source advocate & contributor

About Seiden Group

Team of experts who:

- Mentor IBM i teams, IT Directors, CIOs
- Consult on projects and process
- Develop applications
- Troubleshoot

www.seidengroup.com



Open source languages (In order of “official” appearance on IBM i)

PHP
- Zend + IBM
- 2006

Ruby
- Added with TR7
- 2013

NodeJS
- Option 1
- 5733-OPS/Yum

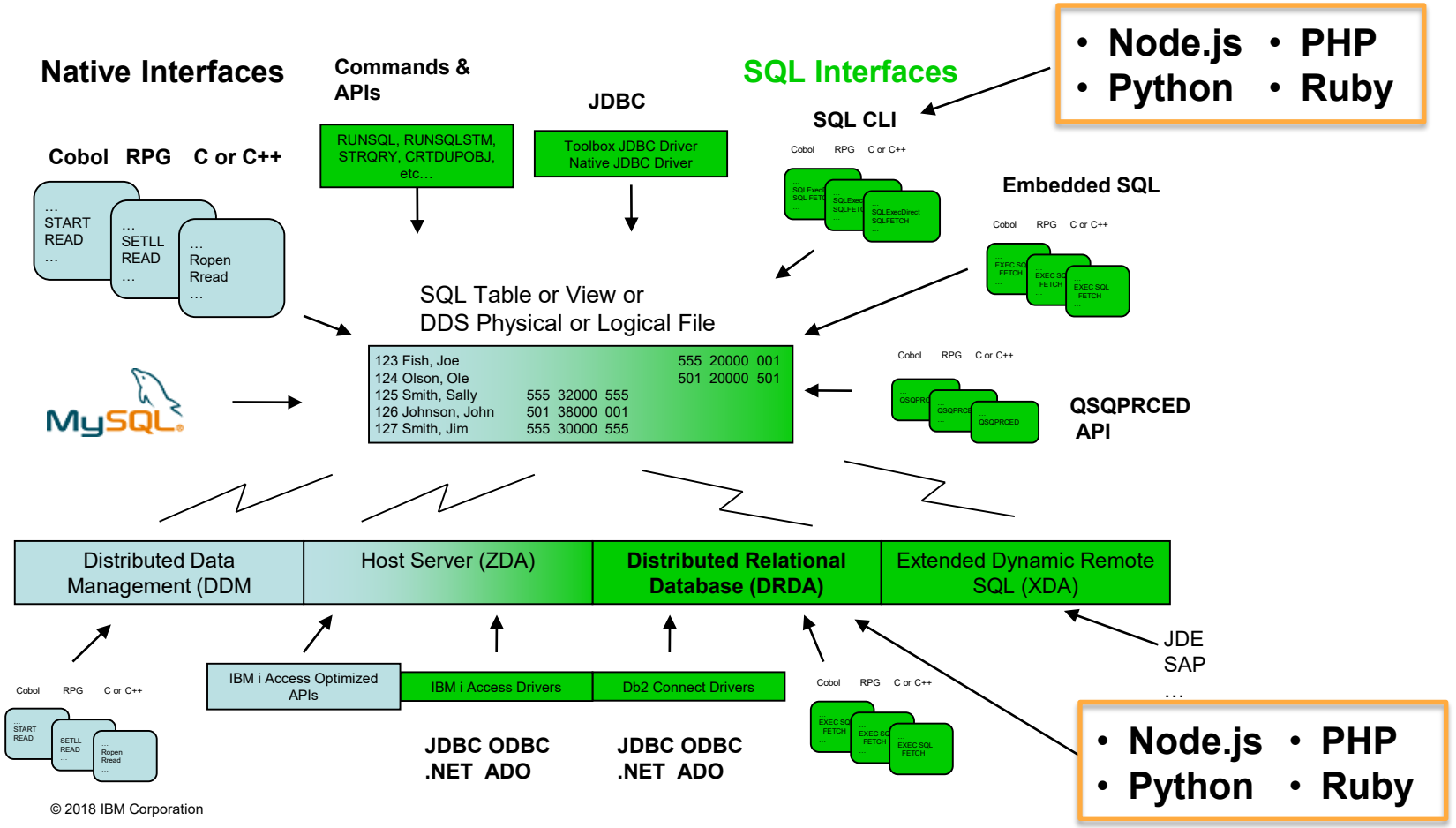
Python
- Option 2
- 5733-OPS/Yum

IBM has integrated Db2 with open source languages

- PHP: ibm_db2
  - https://pecl.php.net/package/ibm_db2
- Python: ibm_db/imb_dbi
  - https://github.com/ibmdb/python-ibmdb
- Ruby: ibm_db
  - https://rubygems.org/gems/ibm_db/versions/3.0.1
- Nods.js: ibm_db
  - https://github.com/ibmdb/node-ibm_db

```
<?php
$pet = array(0, 'cat', 'Pook', 3.2);

$insert = 'INSERT INTO animals (id, breed, name, weight)
    VALUES (?, ?, ?, ?)';

$stmt = db2_prepare($conn, $insert);
if ($stmt) {
    $result = db2_execute($stmt, $pet);
    if ($result) {
        print "Successfully added new pet.";
    }
}
```

Db2 for i – One database, many interfaces



Why should you use (or expand your use of) SQL on i?

- **Strategic interface:**
  - **Strategic database interface for industry**
  - **Strategic interface for IBM i**
  - **Portability of code & skills**
- **Faster delivery on IT requirements**
- **Performance & Scalability**
- **Increased Data Integrity**
- **Easy access to remote databases**
- **IBM i Navigator tooling (both Windows & Access Client Solutions)**

Leverage what you already own

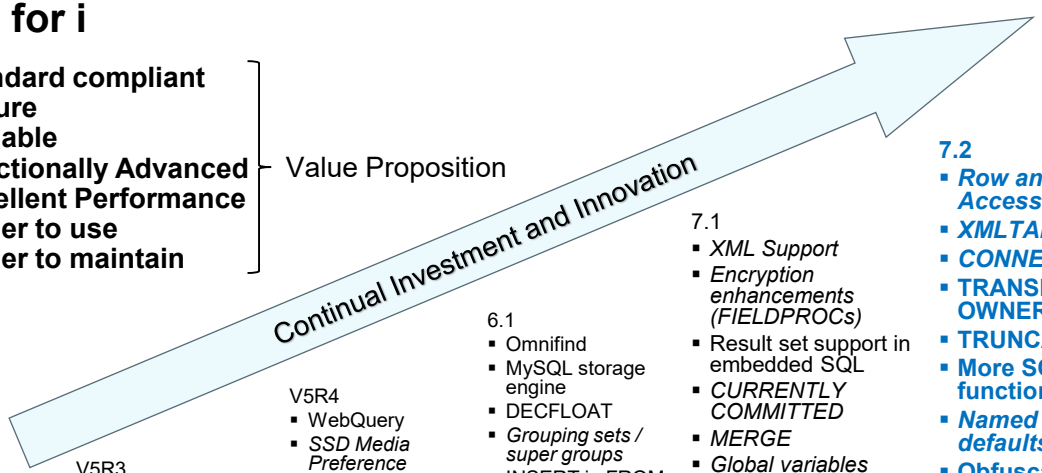
- **State of the art SQL Query Engine (SQE)**
  - Cost-based optimization – determine the most efficient data access methods to fulfill the query
  - Self learning and adapting, based on statistics
  - Query rewrite - rewrite the query to a more efficient equivalent statement
  - Graph based representation of the query

- **SQL Plan Cache – system wide cache of pre-optimized statements, no matter the interface that was used**
- **Global Statistics manager – vital statistics for the cost-based optimizer**
- **IBM i Access Client Solutions – tools for the database user/developer**
  - Index Advisor
  - Show Statements
  - Create Based On
  - SQL Examples and CL prompting
  - SQL Details for jobs
  - Performance Monitoring and Analyze
  - Visual Explain for Queries

Db2 for i

- **Standard compliant**
- **Secure**
- **Scalable**
- **Functionally Advanced**
- **Excellent Performance**
- **Easier to use**
- **Easier to maintain**

Value Proposition



V5R2
- *SQE Stage 1*
- *IASPs*
- Identity columns
- Savepoints
- *UNION in views*
- *Scalar subselect*
- UDTFs
- DECLARE GLOBAL TEMPORARY TABLE
- *Catalog views*

V5R3
- *Partitioned tables*
- UFT-8 & UTF-16
- ICU sort sequence
- MQTs
- Sequences
- Implicit char/numeric
- BINARY / VARBINARY
- GET DIAGNOSTICS
- *DRDA Alias*
- *DECIMAL(63)*
- *SQE Stage 3*
- *Ragged SWA*
- QDBRPLAY
- *Online Reorganize*

V5R4
- WebQuery
- *SSD Media Preference*
- *On Demand Performance Center*
- *Health Center*
- *Completion of SQL Core*
- *Scalar fullselect*
- *Recursive CTE*
- *OLAP expressions*
- INSTEAD OF triggers
- Descriptor area
- *XA support*
- *DDM 2-phase*
- Scrollable cursor
- 2M SQL statement
- 1000 tables in a query

6.1
- Omnifind
- MySQL storage engine
- DECFLOAT
- *Grouping sets / super groups*
- INSERT in FROM
- VALUES in FROM
- Extended Indicator Variables
- Expression in Indexes
- ROW CHANGE TIMESTAMP
- *Statistics catalog views*
- CLIENT special registers
- *SQE Stage 6*
- *DDM and DRDA IPv6*
- *Deferred Restore of MQT and Logicals*
- *Environmental limits*

7.1
- *XML Support*
- *Encryption enhancements (FIELDPROCs)*
- Result set support in embedded SQL
- *CURRENTLY COMMITTED*
- *MERGE*
- *Global variables*
- *Array support in procedures*
- Three-part names and aliases
- SQE Logical file support
- *SQE Adaptive Query Processing*
- *EVI enhancements*
- *Inline functions*
- CREATE OR REPLACE
- *Partition table enhancements*
- Expressions in CALL
- *TR-timed enhancements*

7.2
- ***Row and Column Access Control***
- ***XMLTABLE***
- ***CONNECT BY***
- **TRANSFER OWNERSHIP**
- **TRUNCATE**
- **More SQL Scalar functions**
- ***Named arguments and defaults for parameters***
- **Obfuscation of SQL routines**
- ***Array support in UDFs***
- **Timestamp precision**
- ***Multiple-action Triggers***
- **Built-in Global Variables**
- **1.7 Terabyte Indexes**
- **Navigator Graphs and Charts**
- ***Regular Expressions***
- ***SQL Dynamic Compound***
- **UPDATE ROW across partitions**
- ***System Limits - IFS***

7.3
- **Temporal Tables**
- **Generated columns for auditing**
- ***OLAP Extensions***
- ***OFFSET and LIMIT***
- **Inline User-Defined Table Functions**
- ***New Aggregate Functions***
- **Index Merge Ordering**
- ***EVI Only Access***
- **More Built-in Global Variables**
- **More SQL Scalar functions**
- **Increased routine and view limits**
- ***More Services***
- ***CREATE OR REPLACE TABLE***
- ***ATTACH & DETACH Partition***
- **Pipelined Table Functions**

Enhancements delivered via Db2 PTF Groups

Db2 for i updates by category
Db2 for i Functional Enhancements
Db2 for i Security Enhancements
Db2 for i Performance Enhancements
Db2 for i Database Management Enhancements
Db2 for i Availability/Recovery Enhancements
OmniFind for IBM i
SQL Services

	Database Enhancement Landing Pages
IBM i 7.3	TR3 - TR2 -TR1 - Base Enhancements
IBM i 7.2	TR7 - TR6 -TR5 - TR4 - TR3 - TR2 - TR1 - Base Enhancements
IBM i 7.1	TR11 - TR10 - TR9 - TR8 - TR7

Db2 for i updates by PTF Group and year
Db2 for i PTF Groups - 2018
Db2 for i PTF Groups - 2017
Db2 for i PTF Groups - 2016
Db2 for i PTF Groups - 2015
Db2 for i PTF Groups - 2014
Db2 for i PTF Groups - 2013

www.ibm.com/developerworks/ibmi/techupdates/db2

CREATE TABLE +
Auto Incrementing Row IDs

Autoincrementing (identity) columns

```
CREATE TABLE mylib.best_data_table FOR SYSTEM NAME datatbl
(
  id INTEGER GENERATED ALWAYS AS IDENTITY
  (
         START WITH 1 INCREMENT BY 1
         NO MINVALUE NO MAXVALUE
         NO CYCLE NO ORDER CACHE 20
  ),
  data_column FOR COLUMN datacol VARCHAR(32) NOT NULL,
  more_data_column FOR COLUMN moredata VARCHAR(128) NOT NULL
 CONSTRAINT mylib.datatbl_pk PRIMARY KEY( id )
);
```

CREATE OR REPLACE TABLE

Data Definition Language (DDL) SQL statements that support the optional 'OR REPLACE' clause:

- CREATE OR REPLACE ALIAS
- CREATE OR REPLACE FUNCTION
- CREATE OR REPLACE MASK
- CREATE OR REPLACE PERMISSION
- CREATE OR REPLACE PROCEDURE
- CREATE OR REPLACE SEQUENCE
- **CREATE OR REPLACE TABLE**
- CREATE OR REPLACE TRIGGER
- CREATE OR REPLACE VARIABLE
- CREATE OR REPLACE VIEW

Replacing a table:
- Data-Centric
- Dependent Views & MQTs preserved
- Triggers preserved
- RCAC controls preserved
- Auditing preserved
- Authorizations preserved
- Comments and Labels preserved
- Rows optionally deleted

Knowledge Center

www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_72/db2/rbafzhctabl.htm?lang=en

Article for previous OR REPLACE statements

iprodeveloper.com/database/use-sql-create-or-replace-improve-db2-i-object-management

How to retrieve id value from open source languages

- PHP - function db2_last_insert_id()
  - “Returns the auto generated ID of the last insert query that successfully executed on this connection"
  - http://php.net/manual/en/function.db2-last-insert-id.php

```
/* Checking for single row inserted. */
$stmt = db2_exec($conn, $insertTable);
$ret =  db2_last_insert_id($conn);
```

- Python: `cur._get_last_identity_val()`

- Node: must query the table

Global Variables

Db2 for i Built-in Global Variables

- Use these variables to deploy advanced logic in triggers, RCAC rules, and more

Added with IBM i 7.2 SF99702 Level 3

Available with base IBM i 7.2

Variable name	Schema	Data Type	Size
JOB_NAME	**QSYS2**	**VARCHAR**	**28**
SERVER_MODE_JOB_NAME	**QSYS2**	**VARCHAR**	**28**
CLIENT_IPADDR	SYSIBM	VARCHAR	128
CLIENT_HOST	SYSIBM	VARCHAR	255
CLIENT_PORT	SYSIBM	INTEGER	-
PACKAGE_NAME	SYSIBM	VARCHAR	128
PACKAGE_SCHEMA	SYSIBM	VARCHAR	128
PACKAGE_VERSION	SYSIBM	VARCHAR	64
ROUTINE_SCHEMA	SYSIBM	VARCHAR	128
ROUTINE_SPECIFIC_NAME	SYSIBM	VARCHAR	128
ROUTINE_TYPE	SYSIBM	CHAR	1

Create Variable

- Creates a global variable object which can be used like a host variable or column name.
- The variable is instantiated upon first use within a session. The instantiation can be an assignment statement or a reference. For references, a default value or an expression can be used to supply the value.
- Global variables provide application developers a cost effective alternative to modifying or extending application logic.
- Retrieve from RPG, CL, PHP, etc

***Example**:*

Create a global variable to indicate the department where an employee works.

```
CREATE VARIABLE GV_DEPTNO INTEGER
    DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
    WHERE EMPUSER = SESSION_USER))
```

Global Variable real-world use

- Challenge: configure a dynamic start date using a global variable.

```
CREATE OR REPLACE VARIABLE BHMINSTARTDATE DATE  DEFAULT  '2016-01-01'  ;
```

- Easy testing: change its value in job:

```
set MYSCHEMA.BHMINSTARTDATE = '2014-01-01';
```

Refer to Global Variable in a view

```
MAX(SBSTSTDTA.BHMINSTARTDATE, BFDTTR) AS BFSTDT,
            CASE
            WHEN SBSTSTDTA.BHMINSTARTDATE > CURRENT DATE
                        THEN 0
            ELSE
                        DAYS(CURRENT_DATE) - DAYS(MAX(SBSTSTDTA.BHMINSTARTDATE, BFDTTR))
            END
AS BFIAGE,
```

Data using global variable in view

BFRIPR	BFDSDC	BFDSAM	BFDATE	BFDTTR	BFIAGE	BFSTDT	BFCSBT	BFQTY	BFAMT
	10	21.60	52113	2013-05-21	727	2014-01-01	C	4	270.96
	0	0.00	52113	2013-05-21	727	2014-01-01	C	2	151.92
	10	54.00	13013	2013-01-30	727	2014-01-01	C	10	677.40
	10	10.80	52113	2013-05-21	727	2014-01-01	C	2	135.48
	0	0.00	13013	2013-01-30	727	2014-01-01	C	3	227.88
	10	10.80	13013	2013-01-30	727	2014-01-01	C	2	135.48
	10	10.80	13013	2013-01-30	727	2014-01-01	C	2	135.48
	10	5.40	52113	2013-05-21	727	2014-01-01	C	1	67.74
	10	16.20	13013	2013-01-30	727	2014-01-01	C	3	203.22
	10	16.20	13013	2013-01-30	727	2014-01-01	C	3	203.22
	0	0.00	13013	2013-01-30	727	2014-01-01	C	1	75.96
	0	0.00	13013	2013-01-30	727	2014-01-01	C	1	75.96
	0	0.00	52113	2013-05-21	727	2014-01-01	C	1	75.96
R	5	2754.00	83115	2015-08-31	120	2015-08-31	C	27	11516.58
R	0	0.00	83115	2015-08-31	120	2015-08-31	C	3	1011.24
	2	48.00	73115	2015-07-31	151	2015-07-31	C	4	312.00
	2	48.00	73115	2015-07-31	151	2015-07-31	C	4	312.00
	2	48.00	73115	2015-07-31	151	2015-07-31	C	4	312.00
	10	7.20	61215	2015-06-12	200	2015-06-12	C	3	206.82
	0	0.00	61215	2015-06-12	200	2015-06-12	C	1	75.96
	10	12.00	61215	2015-06-12	200	2015-06-12	C	5	344.70
	10	12.00	61215	2015-06-12	200	2015-06-12	C	5	344.70
	10	24.00	61215	2015-06-12	200	2015-06-12	C	10	689.40
	10	12.00	61215	2015-06-12	200	2015-06-12	C	5	344.70
	0	0.00	61215	2015-06-12	200	2015-06-12	C	1	75.96
	10	4.80	61215	2015-06-12	200	2015-06-12	C	2	137.88
	10	7.20	61215	2015-06-12	200	2015-06-12	C	3	206.82
	10	2.40	61215	2015-06-12	200	2015-06-12	C	1	68.94
	10	12.00	61215	2015-06-12	200	2015-06-12	C	5	344.70
	0	0.00	61215	2015-06-12	200	2015-06-12	C	1	75.96
	10	9.60	61215	2015-06-12	200	2015-06-12	C	4	275.76

With global variable “minimum start date” set to 2014-01-01,

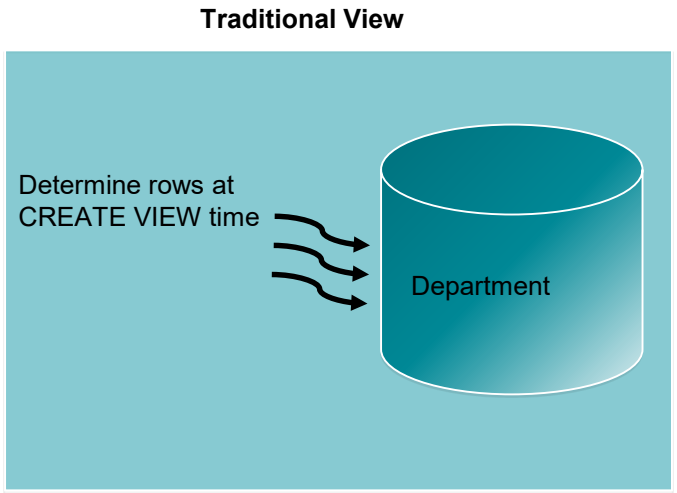
start date values (BFSTDT) less than 1/1/14 appear as 2014-01-01.

Higher values as their actual value. “Age” calculated appropriately.

Views

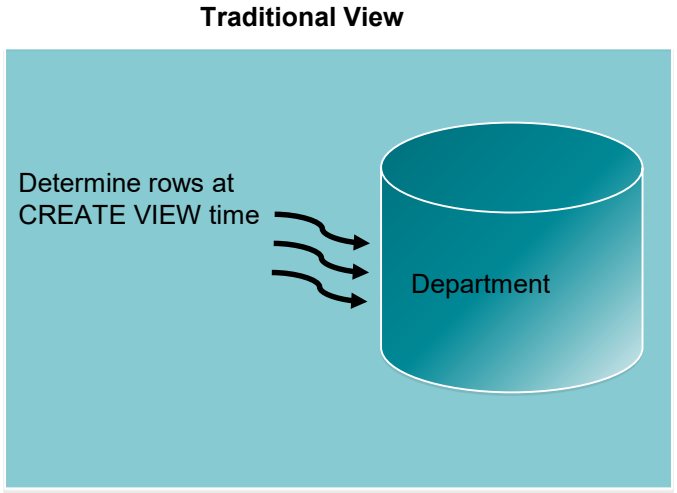
Views

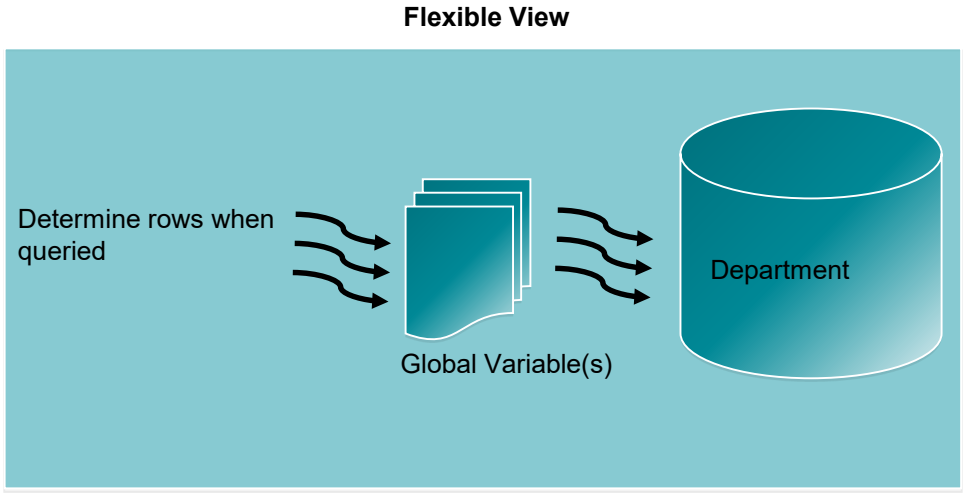
Best Practice:
Don’t allow applications to directly access an SQL Table (physical file)
Why?
Views allow you to limit access to data and more easily change the underlying data model implementation



Flexible Views

- Traditional views are based upon a query that is locked in at create time
- Views with WHERE clause references to Db2 built-in global variables or Db2 global variables are flexible
- With the latest Db2 PTF Group, these views are eligible to be insertable, updateable, and deletable





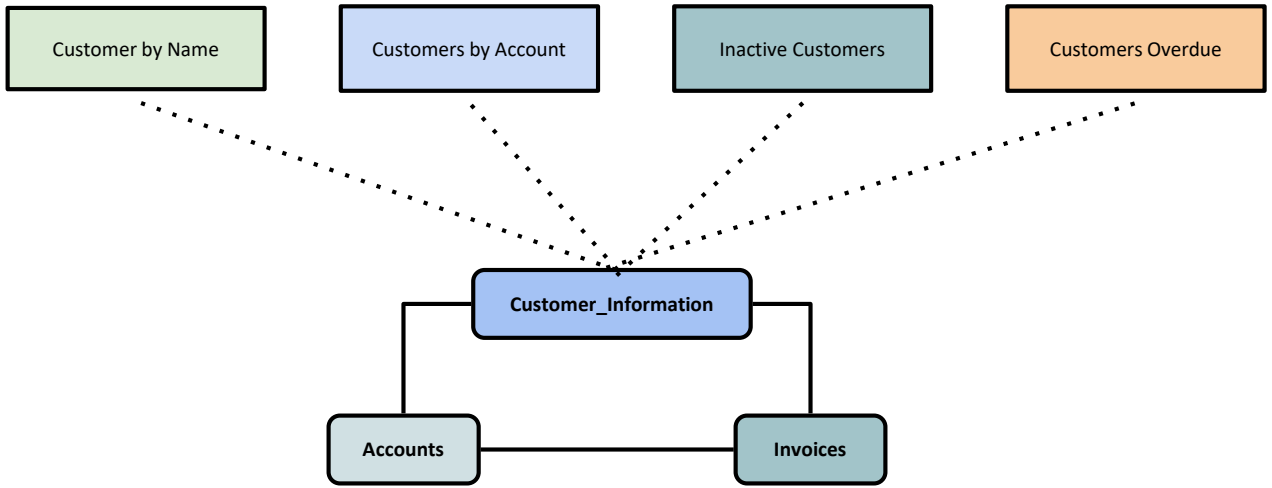
Flexible Views

```
CREATE OR REPLACE VARIABLE TOYSTORE.CURRENT_DEPARTMENT
   FOR SYSTEM NAME CUR_DEPT CHAR(3) DEFAULT  'D21'  ;

CREATE OR REPLACE VIEW TOYSTORE.DEPARTMENT_VIEW FOR SYSTEM NAME
DEPTV AS
SELECT DEPTNO, DEPTNAME, MGRNO , ADMRDEPT, LOCATION FROM
TOYSTORE.DEPARTMENT WHERE TOYSTORE.CURRENT_DEPARTMENT = DEPTNO;

-- Update rows where DEPTNO = 'D21'
UPDATE TOYSTORE.DEPARTMENT_VIEW  SET LOCATION = 'Kingston';

-- Insert a new row
INSERT INTO TOYSTORE.DEPARTMENT_VIEW
VALUES('D33', 'Gardening and landscaping', '000110', 'A00', NULL);
```

Views guarantee consistent data among RPG, PHP, Node…

Modular Db2 logic can be accessed by RPG/PHP/Node/Python, and even other views!

- All accessed through the same view



View with selection logic (WHERE clause)

```
CREATE VIEW MYLIB.ACTIVE_CUSTOMER
AS
        SELECT COMPANY, REGION, SALESREP
        FROM MYLIB.CUSTOMER
        WHERE ACTIVE = 1
```

View joining data from multiple tables

```
CREATE VIEW MYLIB.LOWER_COST
AS
        SELECT SUPPLIER_NUMBER, A.ITEM_NUMBER, UNIT_COST, SUPPLIER_COST
        FROM
                MYLIB.INVENTORY_LIST A
        INNER JOIN
                MYLIB.SUPPLIERS B ON A.ITEM_NUMBER = B.ITEM_NUMBER
        WHERE
                UNIT_COST > SUPPLIER_COST
```

View with casting and calculating logic

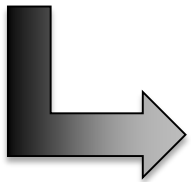
```
-- Note the column name aliasing here
CREATE VIEW MYLIB.LOWER_COST
AS
            SELECT
                        TRIM(SUPPLIER_NUMBER),
                        A.ITEM_NUMBER,
                        DECIMAL(UNIT_COST, 9, 2) AS RETAIL_COST,
                        DECIMAL(SUPPLIER_COST, 9, 2) AS WHOLESALE_COST,
                        DECIMAL(UNIT_COST, 9, 2) - DECIMAL(SUPPLIER_COST, 9, 2) AS MARKUP
            FROM
                        MYLIB.INVENTORY_LIST A
            INNER JOIN
                        MYLIB.SUPPLIERS B ON A.ITEM_NUMBER = B.ITEM_NUMBER
            WHERE
                        UNIT_COST > SUPPLIER_COST;
```

View with user-defined function (more later)

```
create view preslldtj1 (
programcode, programdesc, item, orderbydate, orderbydate_mmddyy,
expired)
as
    select  trim(pccode), trim(pcdesc), pmprod, pmobdate,
    mmddyy_slashes(pmobdate),
    CASE WHEN CURRENT_DATE > PMOBDATE THEN 1 ELSE 0 END as expired
    from preslldt pd left join PRESLLMSTj1 PS on PDPROD = PS.PMPROD
```

Query simplified when complexity is abstracted in view

```
select
     programcode, programdesc, item, orderbydate, orderbydate_mmddyy, expired
from
     preslldtj1
order by
     orderbydate
```



PROGRAMCODE	PROGRAMDESC	ITEM	ORDERBYDATE	ORDERBYDATE_MMDDYY	EXPIRED
HNRK	HENDRICKS GIN PRE-SELL	1306920	2015-06-26	6/26/15	1
HNRK	HENDRICKS GIN PRE-SELL	1306930	2015-06-30	6/30/15	1
HNRK	HENDRICKS GIN PRE-SELL	1306900	2015-06-26	6/26/15	1
BJNV	BEAUJOLAIS NOUVEAU15	6641041	2015-06-21	6/21/15	1
HH15	HEAVEN HILL 2015	1544340	2015-07-05	7/5/15	0
HH15	HEAVEN HILL 2015	1544540	2015-07-04	7/4/15	1
HH15	HEAVEN HILL 2015	1544240	2015-07-05	7/5/15	0
JPIN	JP INSIGNIA 15	8218050	2015-07-02	7/2/15	1

Scalar functions

Trim() solves “trailing blanks” issue

```
$conn = db2_connect(‘*local’, ‘myuser’, ‘mypass’);

$sql = “SELECT mycode FROM account WHERE id = 5”; // mycode is 3A fixed lengh
$stmt = db2_prepare($conn, $sql);
$result = db2_execute($stmt); // params go here if needed
$row = db2_fetch_assoc($stmt);

echo $row[‘MYCODE’];

// MYCODE: 'B  ';

// MYCODE was a fixed-length character string, length 3, so we get trailing spaces

Use VARCHAR in future, but for now, “reface” your database (as Paul Tuohy put it)
$sql = “SELECT trim(mycode) as mycode FROM account WHERE id = 5”;

// MYCODE: 'B';
```

RTRIM() and concatenation

Manipulations in SQL make neater output

```
$sql = ‘SELECT  CAST(RIGHT('000000' CONCAT RTRIM(BFINV#), 6) as char(6) CCSID 37) AS "BFINV#",
CAST(RIGHT('00' CONCAT RTRIM(BFSEQ#), 2) as char(2) CCSID 37) AS "BFSEQ#", CAST(RIGHT('000000’
CONCAT RTRIM(BFDATE), 6) as char(6) CCSID 37) AS "BFDATE", CAST(RIGHT('00' CONCAT
RTRIM(BFLOCB), 2) as char(2) CCSID 37) AS "BFLOCB", trim(BFQTY) AS "BFQTY", trim(BFTYPE) AS
"BFTYPE", trim(BFQSHP) AS "BFQSHP", CAST(RIGHT('000000' CONCAT RTRIM(BFCUST), 6) as char(6)
CCSID 37) AS "BFCUST", CAST(RIGHT('000’ CONCAT RTRIM(BFSLSM), 3) as char(3) CCSID 37) AS
"BFSLSM", CAST(RIGHT('0000000’ CONCAT RTRIM(BFPROD), 7) as char(7) CCSID 37) AS "BFPROD",
trim(CSTRDN) AS "CSTRDN", "C"."CSTERM", trim(CSCITY) AS "CSCITY", trim(P2SIZEDS) AS "P2SIZEDS",
trim(P2LDES) AS "P2LDES", "R"."ARPAID", trim(STBARCD) AS "STBARCD" FROM "BILLHDFL07" AS "B"
       INNER JOIN "CUSTFILE" AS "C" ON B.BFCUST = C.CSCUST
       LEFT JOIN "PRODMST2" AS "P" ON B.BFPROD = P.P2PROD
       LEFT JOIN "ARFILE01L" AS "R" ON ‘   [continues]
$conn = db2_connect(‘*local’, ‘myuser’, ‘mypass’);
$stmt = db2_prepare($conn, $sql);
$result = db2_execute($stmt); // params go here if needed
while ()…
   $row = db2_fetch_assoc($stmt);
}
```

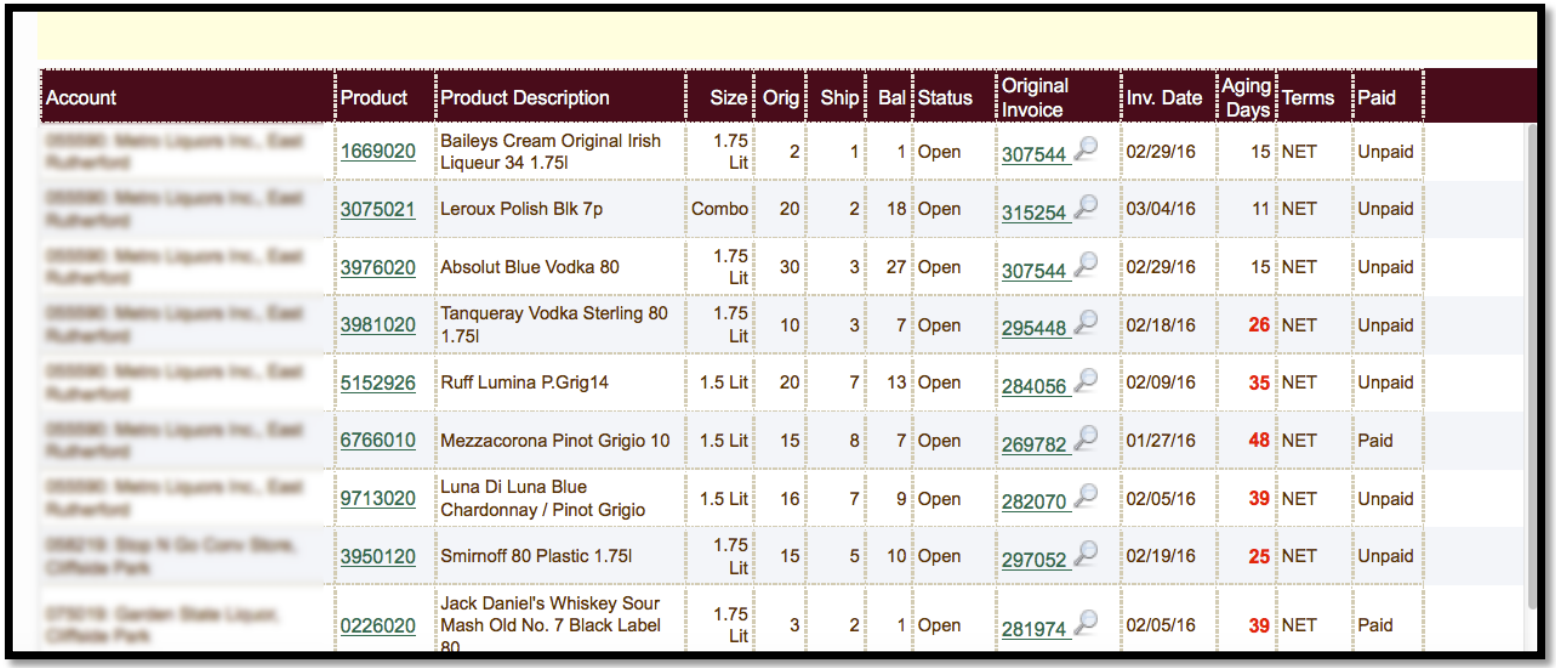
Note the trim() scalar functions

Resulting trimmed/neat JSON

```
$json = json_encode($rows);
```


```
{"identifier":"ROWNUM","items":[{"ROWNUM":"197646","BFLOCB":"05","BFQTY":"20","BFTYPE":"1","BFQSHP":"15"
,"BFSLSM":"219","BFPROD":"0183020","CSTERM":"0","P2LDES":"Seagram Blended Whiskey Seven Crown 80 1.75l"
,"STBARCD":"0466752820645102051602","paid":"Unpaid","shipto":"046675: Bayway Liquors, Elizabeth","rlsqty"
:"","SDESC":"Open","BFBAL":5,"aging":"29","TERMS":"NET","invdate":"2016-02-05","account":"046675: Bayway
 Liquors, Elizabeth","size":"1.75 Lit","invoice":"282064","bfseq":"01"},{"ROWNUM":"200676","BFLOCB":"05"
,"BFQTY":"15","BFTYPE":"1","BFQSHP":"10","BFSLSM":"035","BFPROD":"0200340","CSTERM":"0","P2LDES":"Red
 Stag Spiced","STBARCD":"0466759720265105271502","paid":"Paid","shipto":"046675: Bayway Liquors, Elizabeth"
,"rlsqty":"","SDESC":"Open","BFBAL":5,"aging":"64","TERMS":"NET","invdate":"2015-05-27","account":"046675
: Bayway Liquors, Elizabeth","size":"750 Ml","invoice":"972026","bfseq":"01"},{"ROWNUM":"196089","BFLOCB"
:"05","BFQTY":"40","BFTYPE":"1","BFQSHP":"0","BFSLSM":"219","BFPROD":"0216041","CSTERM":"0","P2LDES"
:"Bay Jack Sgl 15-3315","STBARCD":"0466750982235109101506","paid":"Paid","shipto":"046675: Bayway Liquors
```

How the JSON is used on the page



Account	Product	Product Description	Size	Orig	Ship	Bal	Status	Original Invoice	Inv. Date	Aging Days	Terms	Paid
[redacted]	1669020	Baileys Cream Original Irish Liqueur 34 1.75l	1.75 Lit	2	1	1	Open	307544	02/29/16	15	NET	Unpaid
[redacted]	3075021	Leroux Polish Blk 7p	Combo	20	2	18	Open	315254	03/04/16	11	NET	Unpaid
[redacted]	3976020	Absolut Blue Vodka 80	1.75 Lit	30	3	27	Open	307544	02/29/16	15	NET	Unpaid
[redacted]	3981020	Tanqueray Vodka Sterling 80 1.75l	1.75 Lit	10	3	7	Open	295448	02/18/16	26	NET	Unpaid
[redacted]	5152926	Ruff Lumina P.Grig14	1.5 Lit	20	7	13	Open	284056	02/09/16	35	NET	Unpaid
[redacted]	6766010	Mezzacorona Pinot Grigio 10	1.5 Lit	15	8	7	Open	269782	01/27/16	48	NET	Paid
[redacted]	9713020	Luna Di Luna Blue Chardonnay / Pinot Grigio	1.5 Lit	16	7	9	Open	282070	02/05/16	39	NET	Unpaid
[redacted]	3950120	Smirnoff 80 Plastic 1.75l	1.75 Lit	15	5	10	Open	297052	02/19/16	25	NET	Unpaid
[redacted]	0226020	Jack Daniel's Whiskey Sour Mash Old No. 7 Black Label 80	1.75 Lit	3	2	1	Open	281974	02/05/16	39	NET	Paid

User Defined Functions (UDF)

Functions

- Functions from SQL standpoint are programs, written in either SQL language or external programming language, such as RPG, CL, C or C++
- Functions can be passed input parameters and return either a result set (table functions aka UDTF), or a single value (scalar function aka UDF)
- Databases provide built-in functions for commonly used operations, such as MAX, MIN, etc
- Advantages of functions include :
  - Functions are objects, and therefore access to the function can be granted or revoked – their use can be limited to a subset of users
  - Contents and logic in functions can be changed without affecting the application that invokes the function

Function example

Create an SQL scalar function which computes
Total compensation given a particular employee number

```
CREATE FUNCTION GETCOMP (EMP CHAR(6)) RETURNS DECIMAL(9,2)
RETURN (SELECT (COMM+BONUS+SALARY) FROM EMPLOYEE WHERE EMPNO = EMP)
```

<and is used like this >

```
SELECT GETCOMP('000030') FROM SYSIBM.SYSDUMMY1
```

<result>

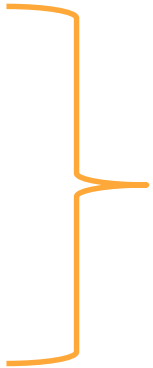
102110.00

Creation time decisions are numerous

Both UDF & UDTF include important create time options.

Taking the default values is not recommended.

Defaults are:
DISALLOW PARALLEL
EXTERNAL ACTION
FENCED
NOT DETERMINISTIC
READS SQL DATA
NOT SECURE



I can’t tell you the correct setting to use, but I can assure that taking the default values is a bad idea

Convert numeric to real date type

UDF definition:

```
CREATE FUNCTION  mmddyy_to_date (thedate numeric(6,0))
            RETURNS DATE LANGUAGE SQL DETERMINISTIC
            BEGIN
                RETURN  date(substr(digits(thedate),1,2) CONCAT '/'
                 CONCAT      substr(digits(thedate),3,2) CONCAT '/'
                 CONCAT      substr(digits(thedate),5,2));
            END
```

Example use:

```
select MMDDYY_TO_DATE(phdate) as podate from podet
inner join pohead on pipo=phpo where piprod = 4317140
```

Format numeric date

Before, with substringing and concatenation in SQL query (DML):

```
select
      date(substr(digits(phdate),1,2) CONCAT '/' CONCAT
      substr(digits(phdate),3,2) CONCAT '/' CONCAT
      substr(digits(phdate),5,2)) as podate
from podet
inner join pohead on pipo=phpo
where piprod = 4317140
```

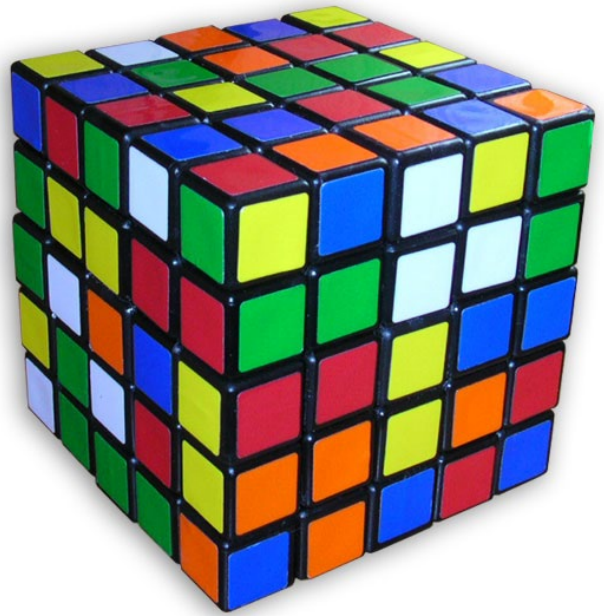
After, create a UDF to handle the formatting (DDL):

```
CREATE FUNCTION  mmddyy_slashes (thedate DATE)
  RETURNS CHAR(8) LANGUAGE SQL DETERMINISTIC
  BEGIN
           RETURN  trim(month(thedate) CONCAT '/' CONCAT
           day(thedate) CONCAT '/’ CONCAT right(year(thedate), 2));
  END
```

Use UDF in a view

```
create view preslldtj1 (
programcode, programdesc, item, orderbydate, orderbydate_mmddyy,
expired)
as
    select  trim(pccode), trim(pcdesc), pmprod, pmobdate,
    mmddyy_slashes(pmobdate),
    CASE WHEN CURRENT_DATE > PMOBDATE THEN 1 ELSE 0 END as expired
    from preslldt pd left join PRESLLMSTj1 PS on PDPROD = PS.PMPROD
```

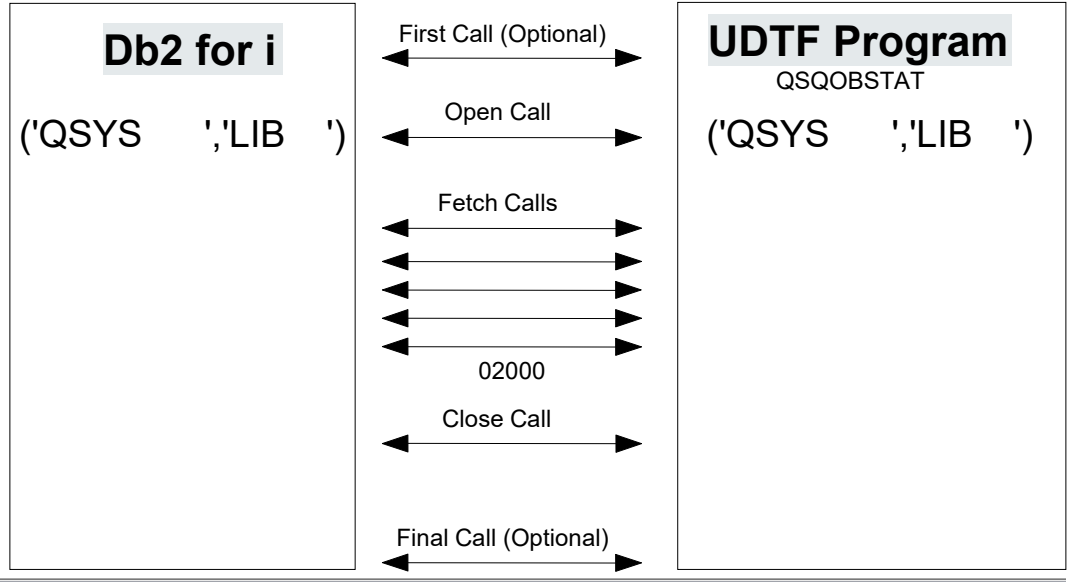
User Defined Table Functions (UDTF)



How Does a Table Function Work?

Your Query

```
SELECT *
FROM TABLE
(qsys2.object_statistics
('QSYS      ','LIB     '))
as x
```



OBJNAME	OBJTYPE	OBJOWNER	OBJDEFINER	OBJCREATED	OBJSIZE	OBJTEXT	OBJLONGNAME
$$D2BS...	*LIB	SAMACKEN	SAMACKEN	2014-10-15 11:23:...	147456	-	$$D2BSYSCF
$$PROD	*LIB	SAMACKEN	SAMACKEN	2014-10-15 11:23:...	73728	-	$$PROD
$AJGOODMA	*LIB	AJGOODMA	AJGOODMA	2015-06-08 10:57:...	155648	Build Library for...	$AJGOODMA
$BLDSHIP	*LIB	SAMACKEN	SAMACKEN	2014-10-15 11:23:...	180224	Build Library for...	$BLDSHIP
$BLDTEST	*LIB	EBERHARD	EBERHARD	2015-08-28 11:22:...	2629632	Build Library for...	$BLDTEST
$CRTBL01	*LIB	SQBATUSER	SQBATUSER	2014-12-05 13:43:...	73728	Test Library crea...	$CRTBL01
$JPBUDIN	*LIB	JPBUDIN	JPBUDIN	2015-06-08 10:57:...	122880	Build Library for...	$JPBUDIN

Challenge: select with complex selection logic

- PHP serving data for mobile app
- Old table structures and relationships made for complex SQL
- RPG was easier for programmer to modify and tune than huge SQL
- RPG (SQL and record-level) eliminated need for complicated JOINs and WHERE clauses and the constant changes to SQL as new requirements were discovered.
- Solution: UDTF presents complex logic as simple table.

SQL to create UDTF:
https://gist.github.com/alanseiden/04235490fe81a1b791ca

UDTF for mobile app

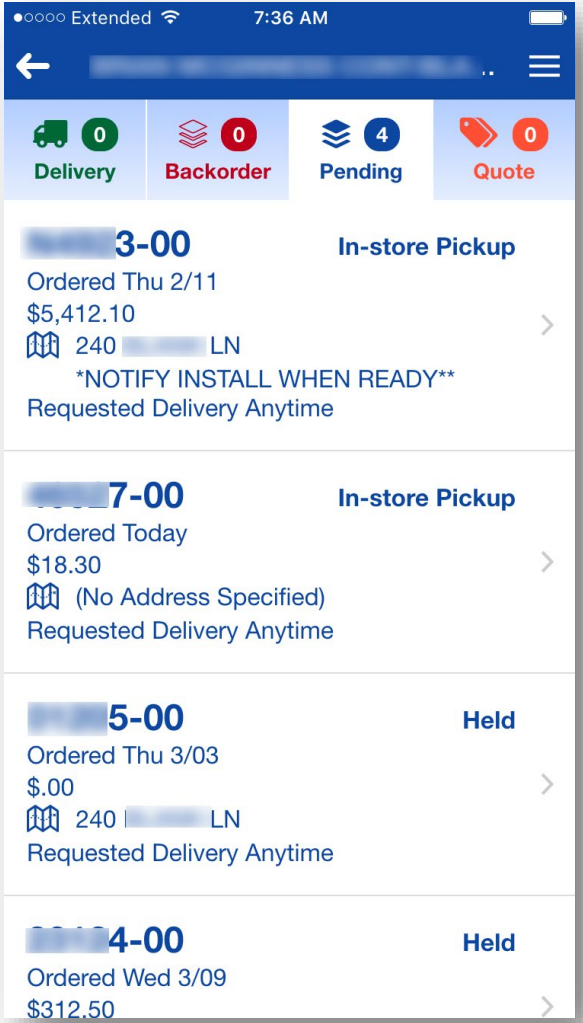
Show daily delivery status of orders in mobile app

Basic flow:

- Native mobile app requests data from PHP on IBM i
- PHP queries Db2 for the data
- PHP converts Db2 data to JSON for app consumption

Challenge:

- False sense of ease from test server with “extracted” data
- To obtain LIVE data on real server, needed to use live tables
- Data in old, poorly formatted tables contained many records
- Selection logic complex using just SQL or Views
- Error-prone and slow!



PHP calling the UDTF named “fnorder”

(We used Zend Framework 2)

```
$orderNumber = 12345; // parameter from app

$sql = "select * from table(fnorder(
                            cast(? as numeric(2,0)),
                            cast(? as char(10))
                        )) as order_details”;

$adapter = $this->getDbAdapter();
$resultHeader = $adapter->query($sqlHeader, array(1, $orderNumber))->toArray();
```

——

After more manipulation in the PHP, JSON looks like this:

https://gist.github.com/alanseiden/e85364ed7515bb6b5bcc

IBM i Services
(access system functions via SQL)

OUTPUT_QUEUE_ENTRIES – the view

Customer:
The QSYS2.OUTPUT_QUEUE_ENTRIES view is too slow!

```
--
-- Find the top 10 consumers of SPOOL storage
--
SELECT USER_NAME, SUM(SIZE) AS total_spool_space
  FROM QSYS2.OUTPUT_QUEUE_ENTRIES
    GROUP BY user_name
    ORDER BY total_spool_space DESC
    LIMIT 10;
```

Scott:
Let's talk about UDTFs!

OUTPUT_QUEUE_ENTRIES – the UDTF

Customer:
That query is 100% faster!

```
--  category:  IBM i Services
--  description:  Spool - Top 10 consumers of spool storage
--
-- Find the top 10 consumers of SPOOL storage
-- Note: Replace library-name with the target library name
--
SELECT user_name, SUM(SIZE) AS total_spool_space
   FROM TABLE(qsys2.object_statistics('QSYS      ', '*LIB')) AS a,
      TABLE(qsys2.object_statistics(a.objname, 'OUTQ')) AS b,
      TABLE(qsys2.output_queue_entries(a.objname, b.objname, '*NO')) AS c
   WHERE user_name NOT LIKE 'Q%'
   GROUP BY user_name
   ORDER BY total_spool_space DESC LIMIT 10;
```

Scott:
Guess what? That query is built into ACS. ☺

IBM® i Services

Security Services

QSYS2.AUTHORITY_COLLECTION – VIEW
QSYS2.AUTHORIZATION_LIST_INFO – VIEW
QSYS2.AUTHORIZATION_LIST_USER_INFO – VIEW
QSYS2.DRDA_AUTHENTICATION_ENTRY_INFO – VIEW
QSYS2.FUNCTION_INFO – VIEW
QSYS2.FUNCTION_USAGE – VIEW
QSYS2.GROUP_PROFILE_ENTRIES – VIEW
QSYS2.OBJECT_PRIVILEGES – VIEW
QSYS2.SQL_CHECK_AUTHORITY – UDF
QSYS2.USER_INFO – VIEW
SYSPROC.SET_COLUMN_ATTRIBUTE – PROCEDURE

Communication Services

QSYS2.NETSTAT_INFO – VIEW
QSYS2.NETSTAT_INTERFACE_INFO – VIEW
QSYS2.NETSTAT_JOB_INFO – VIEW
QSYS2.NETSTAT_ROUTE_INFO – VIEW
QSYS2.SERVER_SBS_ROUTING – VIEW
QSYS2.SET_SERVER_SBS_ROUTING – PROCEDURE
QSYS2.TCPIP_INFO – VIEW
SYSIBMADM.ENV_SYS_INFO – VIEW

Product Services

QSYS2.LICENSE_INFO – VIEW
SYSTOOLS.LICENSE_EXPIRATION_CHECK – PROCEDURE

Application Services

QSYS2.ENVIRONMENT_VARIABLE_INFO – VIEW
QSYS2.QCMDEXC – PROCEDURE
QSYS2.SERVICES_INFO – TABLE
QSYS2.SET_PASE_SHELL_INFO – PROCEDURE

Storage Services

QSYS2.MEDIA_LIBRARY_INFO – VIEW
QSYS2.SYSDISKSTAT – VIEW
QSYS2.SYSTMPSTG – VIEW
QSYS2.USER_STORAGE – VIEW

Journal Services

QSYS2.DISPLAY_JOURNAL – UDTF
QSYS2.JOURNAL_INFO – VIEW

Java Services

QSYS2.JVM_INFO – VIEW
QSYS2.SET_JVM – PROCEDURE

Spool Services

QSYS2.OUTPUT_QUEUE_ENTRIES – VIEW
QSYS2.OUTPUT_QUEUE_ENTRIES – UDTF
QSYS2.OUTPUT_QUEUE_INFO – VIEW

Librarian Services

QSYS2.LIBRARY_LIST_INFO – VIEW
QSYS2.OBJECT_STATISTICS – UDTF

System Health Services

QSYS2.SYSLIMITS – VIEW
QSYS2.SYSLIMTBL – TABLE

Message Handling Services

QSYS2.HISTORY_LOG_INFO – UDTF
QSYS2.JOBLOG_INFO – UDTF
QSYS2.MESSAGE_QUEUE_INFO – VIEW
QSYS2.REPLY_LIST_INFO – VIEW

PTF Services

QSYS2.GROUP_PTF_INFO – VIEW
QSYS2.PTF_INFO – VIEW
SYSTOOLS.GROUP_PTF_CURRENCY – VIEW
SYSTOOLS.GROUP_PTF_DETAILS – VIEW

Work Management Services

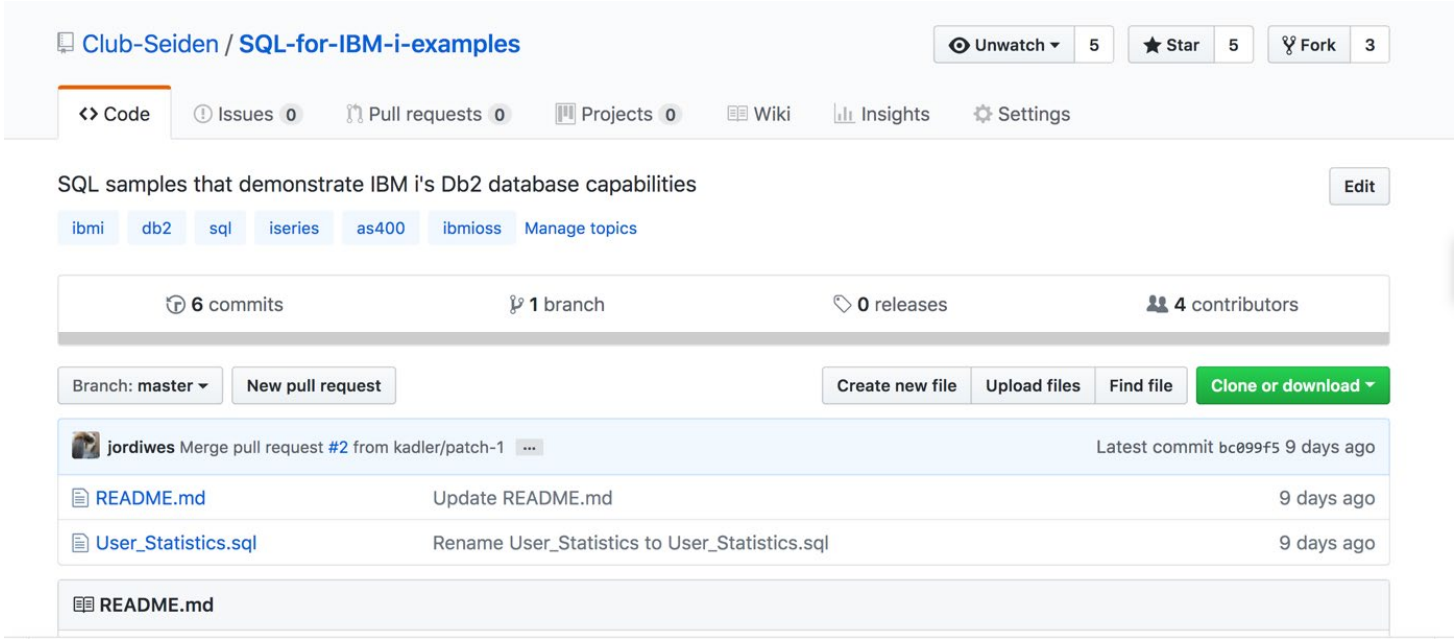
QSYS2.ACTIVE_JOB_INFO – UDTF
QSYS2.GET_JOB_INFO – UDTF
QSYS2.JOB_INFO – UDTF
QSYS2.MEMORY_POOL – UDTF
QSYS2.MEMORY_POOL_INFO – VIEW
QSYS2.OBJECT_LOCK_INFO – VIEW
QSYS2.RECORD_LOCK_INFO – VIEW
QSYS2.SCHEDULED_JOB_INFO – VIEW
QSYS2.SYSTEM_STATUS – UDTF
QSYS2.SYSTEM_STATUS_INFO – VIEW
QSYS2.SYSTEM_VALUE_INFO – VIEW

IBM i Services

http://ibm.biz/DB2foriServices
http://www.ibm.com/developerworks/ibmi/techupdates/db2/landscape

Check out new repository for SQL examples

https://github.com/Club-Seiden/SQL-for-IBM-i-examples



RECORD_LOCK_INFO

- **The Display Record Locks (DSPRCDLCK) command lacks OUTFILE support**
- Implement improved application error handling using SQL
- Use a WHERE clause to avoid a long running query!

```
--
-- What record locks are held over tables in TOYSTORE
--
select * FROM QSYS2.RECORD_LOCK_INFO
where SYSTEM_TABLE_SCHEMA = 'TOYSTORE'
```

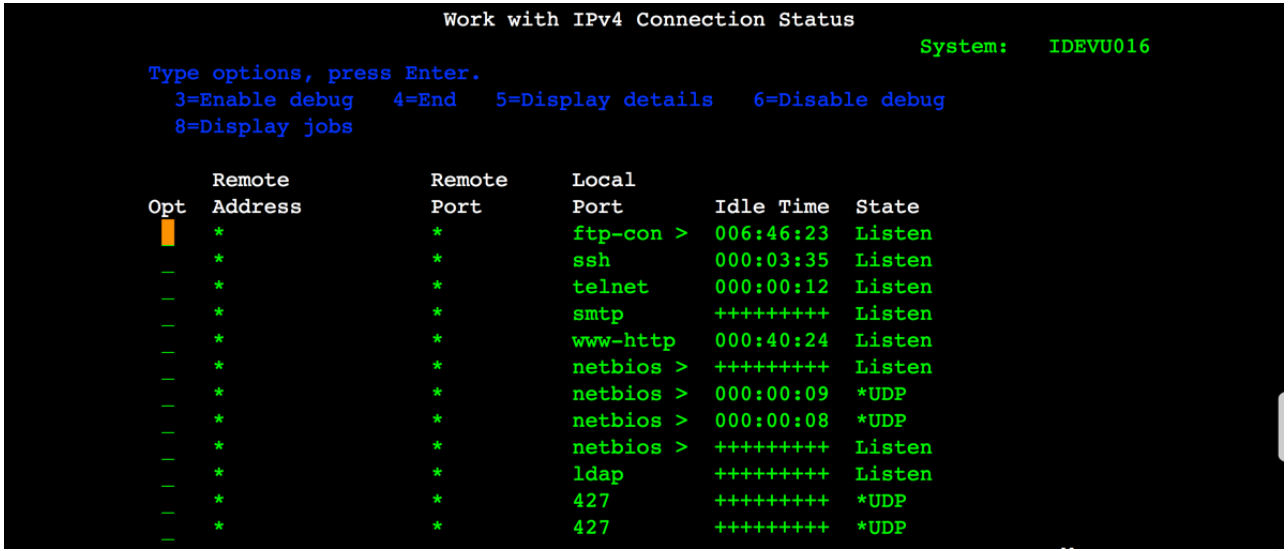
select * FROM QSYS2.RECORD_LOCK_INFO where SYSTEM_TABLE_SCHEMA = 'TOYSTORE' - Sq710(Ur30p58)

TABLE_SCHEMA	TABLE_NAME	TABLE_PARTITION	RELATIVE_RECORD_NUMBER	LOCK_STATE	LOCK_STATUS	JOB_NAME
TOYSTORE	SALES	SALES	1	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	2	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	3	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	4	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	5	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	6	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	7	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	8	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	9	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	10	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	11	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	12	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	13	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	14	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	15	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	16	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	17	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	18	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	19	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	20	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	21	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	22	UPDATE	HELD	233022/QUSER/QZDASOINIT
TOYSTORE	SALES	SALES	23	UPDATE	HELD	233026/SCOTTF/QPADEV0009
TOYSTORE	SALES	SALES	26	UPDATE	HELD	233026/SCOTTF/QPADEV0009
TOYSTORE	SALES	SALES	23	UPDATE	WAITING	233022/QUSER/QZDASOINIT

NETSTAT: Network status information

NETSTAT shows who is connecting to your system via TCP/IP.
“I am in NETSTAT at least daily, often many times per day.”
-- Larry Bolhuis

“NETSTAT *CNN” results on green screen:



Challenge: NETSTAT not easily accessible from PASE

Solution: IBM i services provide NETSTAT via SQL—universal language of IBM i!

From list of services:
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/Db2%20for%20i%20-%20Services

Views for NETSTAT:
QSYS2.NETSTAT_INTERFACE_INFO
QSYS2.NETSTAT_ROUTE_INFO
QSYS2.NETSTAT_JOB_INFO
QSYS2.NETSTAT_INFO

Let’s create a db2 Node.js script to test NETSTAT

```
var express = require('express');
var app = express();

var http = require('http')
var db = require('/QOpenSys/QIBM/ProdData/Node/os400/db2i/lib/db2')

app.get('/', function(req, res) {
  db.init()
  db.conn("*LOCAL")
  db.exec("SELECT LSTNAM, STATE FROM QIWS.QCUSTCDT", function(rs) {
    res.writeHead(200, {'Content-Type': 'text/plain'})
    res.end(JSON.stringify(rs))
  })
  db.close()
})

// Handle passing through the request from our NodeJS Handler
exports.process = function(req, res) {
    app.handle.apply(app, arguments);
}
```

Python example to see if Node.js server is running

We used port 31339

```
$ cd /home/ALAN/netstat_py
$ python3 netstat.py --port 31339
```

Python code to page through NETSTAT data

```
cn = ibm_db.connect('MYIBMI', 'MYUSER', 'MYPW', {})
    sql = '''
SELECT
        REMOTE_ADDRESS, REMOTE_PORT, REMOTE_PORT_NAME,
        LOCAL_ADDRESS, LOCAL_PORT, LOCAL_PORT_NAME,
        CONNECTION_TYPE,
        TRIM(AUTHORIZATION_NAME) AS AUTHORIZATION_NAME, JOB_NAME, SLIC_TASK_NAME
    FROM QSYS2.NETSTAT_JOB_INFO
    {0} -- WHERE CLAUSE
    ORDER BY LOCAL_PORT, LOCAL_ADDRESS, REMOTE_PORT,  REMOTE_ADDRESS
'''
    sql = sql.format("WHERE LOCAL_PORT = ?") if  args.port is not None else sql.format('')
    params = (args.port,) if args.port is not None else None
    if args.limit is not None:
        sql += "\n    LIMIT {0}".format(args.limit)
    if args.offset is not None:
        sql += "\n    OFFSET {0}".format(args.offset)

    netstat_stmt = ibm_db.prepare(cn, sql)
    ibm_db.execute(netstat_stmt, params)
    row = ibm_db.fetch_assoc(netstat_stmt)
    if row:
        rows = [row]
        while row != False:
            rows.append(row)
            row = ibm_db.fetch_assoc(netstat_stmt)
        print(tabulate(rows, 'keys'))
    ibm_db.close(cn)
```

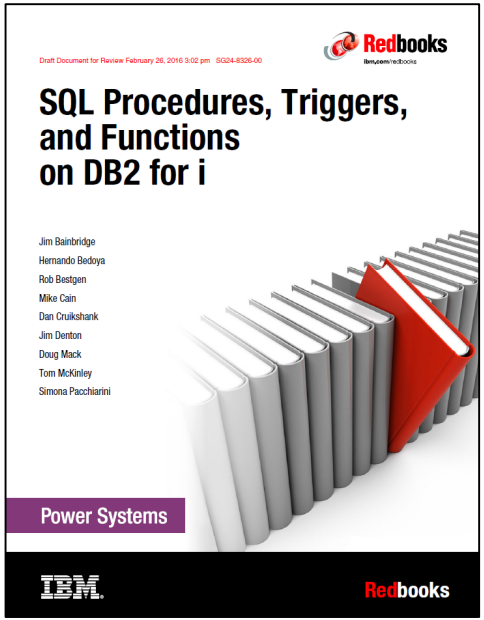
Python NETSTAT data results

```
$ cd /home/ALAN/netstat_py
$ python3 netstat.py --port 31339
SQLConnect(2, IDEVU016, -3, alan, -3, wiscon1, -3)
  LOCAL_PORT_NAME    REMOTE_PORT    SLIC_TASK_NAME  LOCAL_ADDRESS     LOCAL_PORT    REMOTE_PORT_NAME  JOB_NAME                 CONNECTION_TYPE    AUTHORIZATION_NAME    REMOTE_ADDRESS
-----------------  -------------  ----------------  ---------------  ------------  ------------------  -----------------------  -----------------  --------------------  ---------------
                               0                    0.0.0.0                31339                      012422/ALAN/QPADEV000K  IPV4               ALAN                  0.0.0.0
                               0                    0.0.0.0                31339                      012422/ALAN/QPADEV000K  IPV4               ALAN                  0.0.0.0
$
```

Python utilities for IBM i:
https://github.com/Club-Seiden/python-for-IBM-i-examples

Db2 for i – SQL Programming Resources

Essential resource for SQL & Db2 for i database application development



www.redbooks.ibm.com/redpieces/abstracts/sg248326.html

BONUS Reference: database terms

Modern term	Traditional term or phrase
Schema	Library
Table	File or Physical File
Index	Logical File*
Row	Record
Column	Field

* a logical file resembles an “index + view”

Modern views, triggers, etc. are supported by DB2.

Links for open source languages on IBM i

- PHP
  http://www.zend.com/en/products/server/downloads#IBM%20i
- Pythonhttps://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/Python
- Ruby on Rails
  http://powerruby.com/
- Node.js
  http://yips.idevcloud.com/wiki/index.php/NodeJs/NodeJs

Your homework (AKA user group challenge):

Try one of the following (that you haven't tried before)

- UDF – user defined function, like date formatter
- UDTF – user defined table function, like calling an RPG program
- Stored Procedure- very flexible… return data set, insert/update data, call program
- Views – create a view joining two files, or formatting your data the way you like it

Add it to our Github examples repo - https://github.com/Club-Seiden/SQL-for-IBM-i-examples

Contact and tips

Stephanie Rabbani/Alan Seiden

Seiden Group

West Coast/East Coast

Free newsletter:
http://seidengroup.com/tips

Contact us for help with PHP development, Python development, open source training, modernization audit, and much more!

steph@seidengroup.com ● **2507210817** ● **twitter: @jordiwes**

LIMIT and OFFSET

In the days before LIMIT/OFFSET

To do “page at at time” display on web/mobile,

We needed this OLAP syntax:

```
Select
rownumber() over(order by mykey asc) as id,
…
where ID between ? and ?
```

The two ? marks were the starting and ending row numbers to retrieve.

LIMIT and OFFSET

- LIMIT and OFFSET support is popular, but non-standard
  The Db2 Family recently decided to add the support
- This style of data access is most useful for those cases where you only need a subset (page) of rows
- The ***offset-clause*** is only allowed as part of the **outer fullselect** of a DECLARE CURSOR statement or a prepared *select-statement*
- Initially, there is no support in STRSQL

Syntax	Alternative Syntax	Action
LIMIT x	FETCH FIRST x ROWS ONLY	Return the first x rows
LIMIT x OFFSET y	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first **y** rows and return the next **x** rows
LIMIT y , x	OFFSET y ROWS FETCH FIRST x ROWS ONLY	Skip the first **y** rows and return the next **x** rows

OFFSET and LIMIT for Stateless Pagination

Connect,
SELECT…OFFSET 0 LIMIT 5
Fetch 5 rows, Close, Disconnect

Connect,
SELECT…OFFSET 5 LIMIT 5
Fetch 5 rows, Close, Disconnect

Connect,
SELECT…OFFSET 10 LIMIT 5
Fetch 5 rows, Close, Disconnect

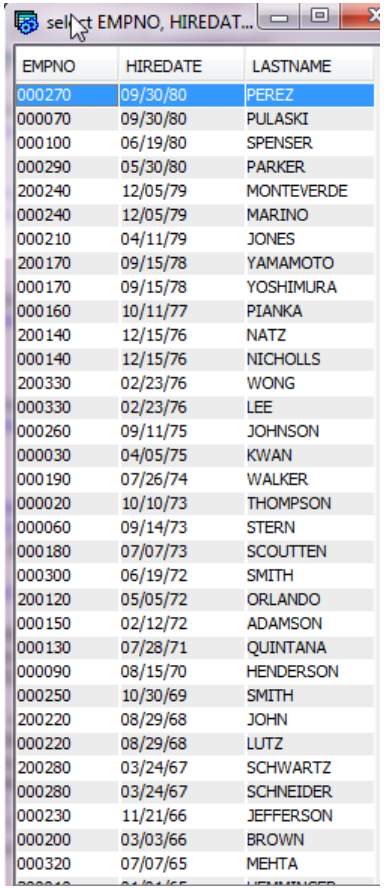
Result set Row Number	Ordering Data	Unique key (Encrypted)
1	Abcd	1234
2	Abdc	3214
3	Acbd	4131
4	Acdb	2143
5	**Bacd**	1243
6	**Bacd**	2341
7	Bcad	4213
8	Bcda	3142
9	Bdac	1423
10	**Bdca**	2431
11	**Bdca**	3412
12	Cadb	1324
13	Cbad	4321

LIMIT and OFFSET

```
CREATE OR REPLACE PROCEDURE TOYSTORE.FIND_EMPLOYEES
  (IN P_PAGESIZE BIGINT, IN P_OFFSET BIGINT)
            DYNAMIC RESULT SETS 1
            LANGUAGE SQL
BEGIN
  DECLARE V_PREP_STMT1 VARCHAR(4096)  ;
  DECLARE CEMP_RESULT_SET1 CURSOR
     WITH RETURN FOR PREP_STMT1;
  SET V_PREP_STMT1 =
  'SELECT EMPNO, HIREDATE, LASTNAME FROM
     TOYSTORE.EMPLOYEE
     ORDER BY HIREDATE DESC
     LIMIT ? OFFSET ?';
  PREPARE PREP_STMT1 FROM V_PREP_STMT1 ;
  OPEN CEMP_RESULT_SET1 USING P_PAGESIZE, P_OFFSET;
END;

CALL TOYSTORE.FIND_EMPLOYEES(10, 0);
CALL TOYSTORE.FIND_EMPLOYEES(10, 10);
```

Page 1

Page 2



EMPNO	HIREDATE	LASTNAME
000270	09/30/80	PEREZ
000070	09/30/80	PULASKI
000100	06/19/80	SPENSER
000290	05/30/80	PARKER
200240	12/05/79	MONTEVERDE
000240	12/05/79	MARINO
000210	04/11/79	JONES
200170	09/15/78	YAMAMOTO
000170	09/15/78	YOSHIMURA
000160	10/11/77	PIANKA
200140	12/15/76	NATZ
000140	12/15/76	NICHOLLS
200330	02/23/76	WONG
000330	02/23/76	LEE
000260	09/11/75	JOHNSON
000030	04/05/75	KWAN
000190	07/26/74	WALKER
000020	10/10/73	THOMPSON
000060	09/14/73	STERN
000180	07/07/73	SCOUTTEN
000300	06/19/72	SMITH
200120	05/05/72	ORLANDO
000150	02/12/72	ADAMSON
000130	07/28/71	QUINTANA
000090	08/15/70	HENDERSON
000250	10/30/69	SMITH
200220	08/29/68	JOHN
000220	08/29/68	LUTZ
200280	03/24/67	SCHWARTZ
000280	03/24/67	SCHNEIDER
000230	11/21/66	JEFFERSON
000200	03/03/66	BROWN
000320	07/07/65	MEHTA

Enhanced – IBM i Services

```
SELECT * from SYSTOOLS.GROUP_PTF_CURRENCY
     WHERE PTF_GROUP_RELEASE = 'R710'
     ORDER BY ptf_group_level_available-ptf_group_level_installed DESC
```

SELECT * from SYSTOOLS.GROUP_PTF_CURRENCY WHERE PTF_GROUP_RELEASE = 'R710' ORDER BY ptf_gr ... - Lp13ut16.rch.stglabs.ibm.com(Lp13ut16)

PTF_GROUP_CURRENCY	PTF_GROUP_ID	PTF_GROUP_TITLE	PTF_GROUP_LEVEL_INSTALLED	PTF_GROUP_LEVEL_AVAILABLE	PTF_GROUP_L
UPDATE AVAILABLE	SF99710	Current Cumulative PTF...	13298	15142	06/24/2015
UPDATE AVAILABLE	SF99709	710 Group Hiper	139	148	09/11/2015
UPDATE AVAILABLE	SF99708	710 Group Security	40	45	09/11/2015
UPDATE AVAILABLE	SF99701	710 DB2 for IBM i	34	37	08/29/2015
UPDATE AVAILABLE	SF99368	710 IBM HTTP Server for i	33	35	07/01/2015
UPDATE AVAILABLE	SF99572	710 Java	21	22	08/28/2015
INSTALLED LEVEL IS CURRENT	SF99367	710 TCP/IP PTF	8	8	06/02/2015
INSTALLED LEVEL IS CURRENT	SF99707	710 Technology Refresh	10	10	05/29/2015

```
SELECT * FROM SYSTOOLS.GROUP_PTF_DETAILS
  WHERE PTF_STATUS <> 'PTF APPLIED' ORDER BY PTF_GROUP_NAME
```

SELECT * FROM SYSTOOLS.GROUP_PTF_DETAILS WHERE PTF_STATUS <> 'PTF APPLIED' ORDER BY PT ... - Lp13ut16.rch.stglabs.ibm.com(Lp13ut16)

PTF_GROUP_DESCRIPTION	PTF_GROUP_NAME	PTF_STATUS	PTF_PRODUCT_ID	PTF_IDENTIFIER	APAR_NAME	PTF_INCLUDED_IN_GROUP_DATE
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5733CY3	SI41186	SE44942	11/30/11
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5733OPS	SI57304	SE62283	07/01/15
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5733WQX	SI56804	SE61892	07/01/15
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5733WQX	SI56889	SE61921	07/01/15
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5770DG1	SI57078	SE62051	07/01/15
IBM HTTP SERVER FOR I...	SF99368	PTF MISSING	5770DG1	SI57111	SE61671	07/01/15

Stored Procedures

Procedures

- Procedures from SQL standpoint are programs, written in either SQL language or external programming language, such as C or C++
- Procedures can be passed input parameters and can return output parameters. Parameters can also be both input and output
- Procedures can return result sets
- Advantages of procedures include :
  - Procedures are objects, and therefore access to the procedures can be granted or revoked – their use can be limited to a subset of users
  - Contents and logic in procedures can be changed without affecting the application that calls the procedure
  - In web environment or client/server, traffic is reduced to a CALL statement, instead of sending statements one at a time

SQL Procedure – Example

```
CREATE PROCEDURE GET_EMPLOYEE (IN PARM1 CHAR(6)) LANGUAGE SQL
BEGIN
DECLARE C1 CURSOR WITH RETURN FOR SELECT * FROM EMPLOYEE WHERE
  EMPNO = PARM1;
OPEN C1;
END
```

<and, to call the procedure>

```
CALL GET_EMPLOYEE ('000030')
```

<returns this result set>

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000030	SALLY	A	KWAN	C01	4738	04/05/2005	MANAGER	20	F	05/11/1971	98250.00	800.00	3060.00

Stored procedure with multiple actions (insert, update…)

```
CREATE OR REPLACE PROCEDURE WEBLIB.GETINVOICE (  IN DAY_ID BIGINT DEFAULT  0  )

LANGUAGE SQL SPECIFIC WEBLIB.GETINVOICE NOT DETERMINISTIC MODIFIES SQL DATA

BEGIN

      INSERT INTO WEBLIB.INVRPF (
            ROWID,  INVOICE_NO, SHIPMENT_NO, CONSIGNEE, SHIPPER,
            CHARGE_CODE, AMOUNT, RATE, PIECES, PAYMENT_TYPE
      ) SELECT * FROM TABLE ( WEBLIB.GETXMLBYID ( DAY_ID ) ) A ;

            INSERT INTO WEBLIB.INVHPF (DAY_ID , INVOICE_NO)
     SELECT DISTINCT ROWID, INVOICE_NO
 FROM WEBLIB.INVRPF
 WHERE ROWID = DAY_ID;

( Continued… )
```

Stored procedure (cont)

```
    UPDATE WEBLIB.INVRPF AR SET
          HROWID = (
                SELECT AH.ROWID
                FROM WEBLIB.ARMINVHPF AH
                WHERE AR.AIROWID = AH.ROWID AND AR.INVOICE_NO = AH.INVOICE_NO
          )
           WHERE ROWID = DAY_ID;

    INSERT INTO WEBLIB.ARMINVCPF (AHROWID , CODE , CHARGE)
           SELECT AHROWID,  CODE, CHARGE
    FROM WEBLIB.ARMINVRPF
    WHERE AIROWID = DAY_ID AND CHARGE_CODE IS NOT NULL ;

           (Repeat for line data)

END;
```