AUGUST 2020 IBMSYSTEMSMAG.COM

Power Systems

IBM Systems

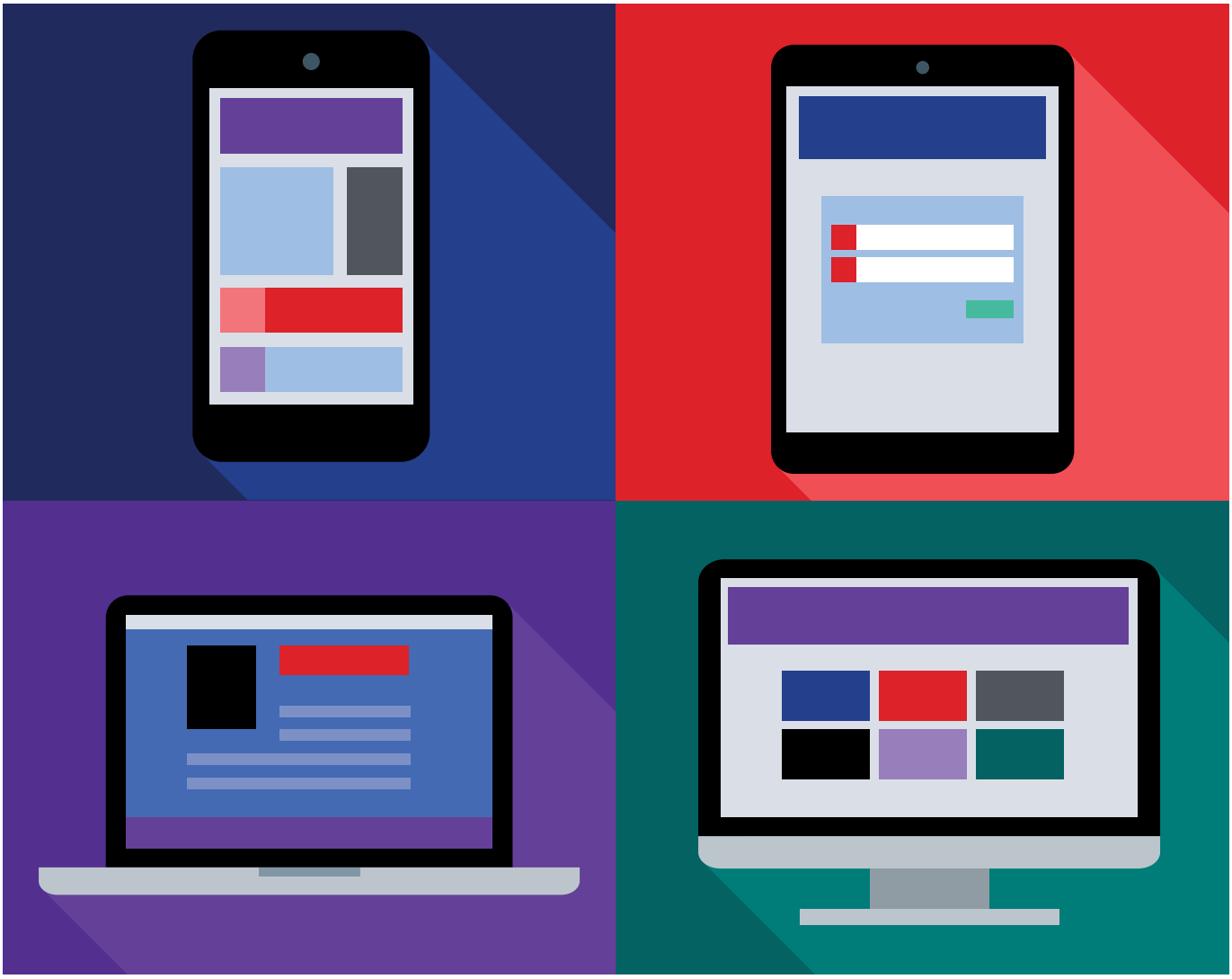
Why IBM POWER9 adds up to big business value for clients

PAGE 18

The Way Forward

Hybrid multicloud enables agile responses to new demands

PAGE 14



Modernize Mission-Critical Applications

With the ODBC driver, IBM i clients can develop applications on a local desktop, then deploy on IBM i

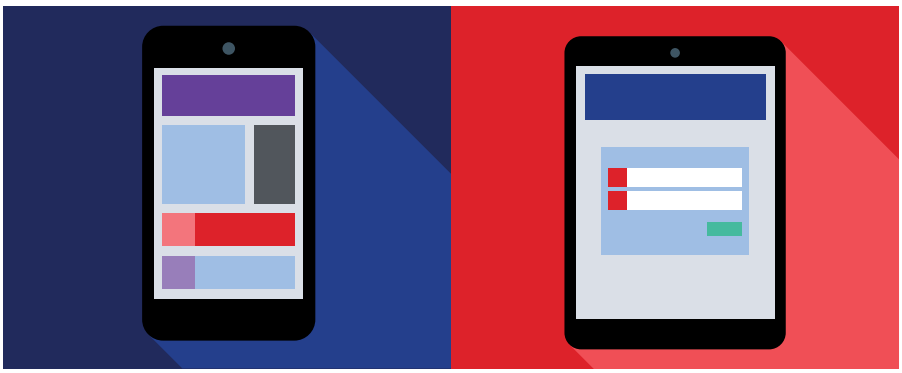
By Scott McKinney

"There are lots of interfaces into the database from the ILE point of view, but from a PASE point of view, we've always had some pain points with the interfaces," says Kevin Adler, advisory software engineer for open source, PASE and IBM i Access ODBC. "We ported the ODBC driver into PASE to make it a little nicer and easier for customers to take advantage of the database from those interfaces and languages."

The driver allows IBM i clients to leverage ODBC's standard interface across multiple databases and platforms, which is easier to work with than a specialized connector written for IBM Db2® specifically. Clients can leverage new environments or new languages quickly, as the driver uses existing ODBC interfaces for those languages. These include Python, Node.js, PHP, Perl, Ruby and any of the new languages on the horizon.

What's Changed

Prior to this, the ODBC driver ported into IBM i. If a developer wanted to work on a local desktop, then deploy that application on IBM i, they'd need Db2 Connect on the desktop. That requires an expensive license, however.



"With the ODBC driver, we can add configuration options and applications can adjust their configuration to take advantage of that, while with the CLI, we have to create connection or statement attributes that applications can set."

–Kevin Adler, advisory software engineer for open source, PASE and IBM i Access ODBC

With the ODBC driver, the interface you use to connect from Linux or Windows® is the same as the one you would use in PASE. They share parameters and connection options and, for the most part, they run the same. "A lot of clients have asked for this, so we've enabled an option that wasn't there before," Adler says.

Increased Performance for Clients

The ODBC driver offers many benefits for IBM i clients, including a built-in configuration capability. In the new environment, clients can use the ODBC driver and a language interface for ODBC, such as PDO_ODBC in PHP or PyODBC in Python. This combination allows you to pass in the data source name (DSN) into an .ini file, and specify configuration options and connection string keywords directly in that .ini file.

"This nice configuration capability allows us to be more responsive to user needs and requirements. With the ODBC driver, we can add configuration options and applications can adjust their configuration to take advantage of that, while with the CLI, we have to create connection or statement attributes that applications can set," says Adler. "Setting connection and statement attributes requires application changes, while adjusting a DSN configuration requires no application changes. That's the benefit the ODBC driver brings."

As the new PHP rolls out, the community of PHP RPMs being provided by Zend don't come with the IBM Db2 and PDO_IBM interfaces that clients typically would have used with Zend Server. Instead, it comes with the ODBC and PDO_ODBC interfaces. "Some of our business and community partners

are seeing performance enhancements as they migrate clients from Zend Server with IBM Db2 to the new PHP with ODBC," says Adler.

Moreover, the ODBC driver means more time for Adler's team to develop other tools, such as an interface into SQLAlchemy, a relational mapper for Python. The team is leveraging the PyODBC interface so it can focus on enhancing the rest of the ecosystem. It completed similar work on LoopBack and Sequelize connectors for IBM i, which are somewhat equivalent but for Node.js.

Maximizing the ODBC Driver

The ODBC driver communicates through the database host server. Unlike CLI, which connects through the PASE API directory memory, you have to make a TCP connection to the host server, and every connection has a prestart job associated with it.

You can use various tuning parameters to optimize prestart jobs for performance. If you figure out your usual or peak loads, you can ensure jobs won't just sit there, taking up resources. To learn more about these parameters, visit the Knowledge Center (ibm.co/3fvvbP1).

Another thing you can do to improve performance is use connection pooling. The ODBC API has connection pooling in its API. Usage and configuration depends on the platform and language interface.

Installing the ODBC Driver

The process for installing is well documented. The driver is available for Windows, Linux, PASE for IBM i, and most recently macOS. When installed on PASE, it comes with a default *LOCAL DSN. Once installed, your language of choice should have an ODBC interface available: PyODBC can be used with Python and is available through IBM's open source package management; node-odbc (which is maintained by IBM) is available to install via npm on Node.js 8 or higher; and the Community PHP offering from Zend comes with both the ODBC and PDO_ODBC interfaces.

"We've definitely improved the experience in out-of-box being able to use the interfaces from all of these languages." says Adler. "We're working on shipping the driver as part of our Yum repository so you can install from the command line or the ACS Open Source Package GUI."

Adler's team will continue to maintain the existing interfaces, but are not going to be actively developing them.

"We're focusing on leveraging the ODBC driver with the language interfaces that are provided for ODBC," Adler says. "I suggest converting existing applications to it, Node or Python interfaces, and definitely recommend you look at using the ODBC driver going forward."

CONFIGURATION OPTIONS WITH CONNECTION STRING KEYWORDS

Some of the connection string options developers ask IBM's Kevin Adler about include:

- **Commitment Control: If you set CommitMode = 0 in the DSN or on the connection string, that will turn off commitment control and use *NONE**
- **DefaultLibraries specifies a comma separated list of libraries when you connect**
- **Naming allows you to set either system naming or SQL naming; it defaults to SQL naming**
- **BlockFetch allows you to fetch blocks at a time into an internal buffer, then reads records out of that block data, thus avoiding network latency from doing lots of row-by-row fetches. The default is set to 1, which enables blocking on fetches of 1 row.**
- **BlockSizeKB sets the block size on fetch requests. It defaults to fetching 256 KB at a time and can specify up to 8,192 KB.**
- **MaxFieldLength sets the maximum large object size that is fetched in line in a row. You can adjust it so that you get a big block of data in one chunk instead of having a lot of roundtrips to the server. It defaults to 32 KB.**

Learn more about connection string options at ibm.co/2Bh6SCa