Artigo
Danusa Calixto · Out. 10, 2022 8min de leitura

CI/CD com SQL do IRIS

No vasto e variado mercado de banco de dados SQL, o InterSystems IRIS se destaca como uma plataforma que vai muito além do SQL, oferecendo uma experiência multimodelo otimizada e a compatibilidade com um rico conjunto de paradigmas de desenvolvimento. Em especial, o mecanismo Object-Relational avançado ajudou as organizações a usar a abordagem de desenvolvimento mais adequada para cada faceta das cargas de trabalho com muitos dados, por exemplo, fazendo a ingestão de dados por objetos e consultando-os simultaneamente por SQL. As Classes Persistentes correspondem às tabelas SQL, suas propriedades às colunas da tabela, e a lógica de negócios é facilmente acessada usando as Funções Definidas pelo Usuário ou os Procedimentos Armazenados. Neste artigo, focaremos um pouco na mágica logo abaixo da superfície e discutiremos como isso pode afetar suas práticas de desenvolvimento e implantação. Essa é uma área do produto em que temos planos de evoluir e melhorar, portanto, não hesite em compartilhar suas opiniões e experiências usando a seção de comentários abaixo.

Salvando a definição de armazenamento

Escrever uma nova lógica de negócios é fácil e, supondo que você tenha APIs e especificações bem definidas, adaptá-la ou ampliá-la também costuma ser. No entanto, quando não é apenas lógica de negócios, mas também envolve dados persistentes, qualquer coisa que você alterar na versão inicial precisará ser capaz de lidar com os dados que foram ingeridos por essa versão anterior.

No InterSystems IRIS, os dados e código coexistem em um único mecanismo de alto desempenho, sem a meia dúzia de camadas de abstração que você vê em outras estruturas de programação 3GL ou 4GL. Isso significa que há apenas um mapeamento muito fino e transparente para traduzir as propriedades da sua classe para posições $list em um nó global por linha de dados ao usar o armazenamento padrão. Se você adicionar ou remover propriedades, não quer que os dados de uma propriedade removida apareçam em uma nova propriedade. É desse mapeamento das propriedades da sua classe que a Definição de Armazenamento cuida, um bloco de XML um pouco enigmático que você deve ter percebido na parte inferior da definição da sua classe. Na primeira vez que você compila uma classe, uma nova Definição de Armazenamento é gerada com base nas propriedades e nos parâmetros da classe. Quando você faz alterações na definição da classe, no momento da recompilação, essas alterações são reconciliadas com a Definição de Armazenamento existente e alteradas para manter a compatibilidade com os dados existentes. Assim, enquanto você se esforça para refatorar as classes, a Definição de Armazenamento considera cuidadosamente sua criatividade anterior e garante que os dados antigos e novos permaneçam acessíveis. Chamamos isso de **evolução de esquema**.

Na maioria dos outros bancos de dados SQL, o armazenamento físico das tabelas é muito mais opaco, se visível, e as alterações só podem ser feitas por declarações ALTER TABLE. Esses são comandos de DDL (linguagem de definição de dados) padrão, mas normalmente são muito menos expressivos do que é possível alcançar ao modificar uma definição de classe e um código de procedimento diretamente no IRIS.

Na InterSystems, nos esforçamos para oferecer aos desenvolvedores do IRIS a capacidade de separar de forma limpa o código e os dados, pois isso é crucial para garantir o empacotamento e a implantação suave dos aplicativos. A Definição de Armazenamento desempenha uma função única nisso, pois captura como um mapeia para o outro. Por isso, vale a pena examinar mais a fundo o contexto de práticas gerais de desenvolvimento e pipelines de CI/CD em particular.

Exportando para UDL

No século atual, o gerenciamento de código-fonte é baseado em arquivos, então vamos primeiro analisar o

formato principal de exportação de arquivos do IRIS. A **Linguagem de Descrição Universal** (Universal Description Language ou UDL, na sigla em inglês) pretende, como o nome sugere, ser um formato de arquivo universal para todo e qualquer código que você escrever no InterSystems IRIS. É o formato de exportação padrão ao trabalhar com o plug-in VS Code ObjectScript e leva a arquivos fáceis de ler que parecem quase iguais ao que você veria em um IDE, com um arquivo .cls individual para cada classe (tabela) no seu aplicativo. Você pode usar $SYSTEM.OBJ.Export() para criar arquivos UDL explicitamente ou apenas aproveitar a integração do VS Code.

Da época do Studio, talvez você se lembre de um formato XML que capturava as mesmas informações do UDL e permitia agrupar várias classes em uma única exportação. Embora essa última parte seja conveniente em alguns cenários, é muito menos prático ler e rastrear diferenças entre versões, então vamos ignorá-la por enquanto.

Como a UDL é destinada a capturar tudo o que o IRIS pode expressar sobre uma classe, ela incluirá todos os elementos de uma definição de classe, incluindo a Definição de Armazenamento completa. Ao importar uma definição de classe que já inclui uma Definição de Armazenamento, o IRIS verificará se essa Definição de Armazenamento abrange todas as propriedades e índices da classe e, se for o caso, usará no estado em que está e substituirá a Definição anterior para essa classe. Isso torna a UDL um formato prático para o gerenciamento de versões das classes e a Definição de Armazenamento, pois preserva essa compatibilidade para dados ingeridos por versões anteriores da classe, onde quer que você implante.

Se você é um desenvolvedor hardcore, talvez se pergunte se essas Definições de Armazenamento continuam crescendo e se essa "bagagem" precisa ser transportada indefinidamente. O objetivo das Definições de Armazenamento é preservar a compatibilidade com dados pré-existentes, portanto, se você sabe que não há nada disso e quiser se livrar de uma genealogia longa, pode "redefinir" a Definição de Armazenamento ao removê-la da definição da classe e gerar outra com o compilador da classe. Por exemplo, você pode usar isso para aproveitar novas práticas recomendadas, como o uso de Conjuntos de Extensão, que implementam nomes globais com hash e separam cada índice em um próprio global, melhorando as eficiências de baixo nível. Para a compatibilidade com versões anteriores nos aplicativos dos clientes, não podemos mudar universalmente esses padrões na superclasse %Persistent (embora os aplicaremos ao criar uma tabela do zero usando o comando DDL CREATE TABLE). Portanto, uma revisão periódica das classes e do armazenamento vale a pena. Também é possível editar o XML de Definição de Armazenamento diretamente, mas os usuários precisam ter muito cuidado, pois isso pode tornar os dados existentes inacessíveis.

Até aqui, tudo bem. As Definições de Armazenamento oferecem um mapeamento inteligente entre suas classes e se adaptam automaticamente com a evolução do esquema. O que mais tem lá?

Estático x Estatísticas?

Como você provavelmente sabe, o mecanismo SQL do InterSystems IRIS faz o uso avançado de estatísticas de tabela para identificar o plano de consulta ideal para qualquer declaração executada pelo usuário. As estatísticas de tabela incluem métricas sobre o tamanho de uma tabela, como os valores são distribuídos em uma coluna e muito mais. Essas informações ajudam o otimizador de SQL do IRIS a decidir qual índice é mais vantajoso, em que ordem unir as tabelas, etc. Portanto, intuitivamente, quanto mais atualizadas as estatísticas estiverem, maiores serão as chances de planos de consulta ideais. Infelizmente, até a introdução da amostragem de bloco rápida no IRIS 2021.2, coletar estatísticas de tabela precisas costumava ser uma operação computacionalmente cara. Portanto, quando os clientes implantavam o mesmo aplicativo em vários ambientes com padrões de dados basicamente iguais, fazia sentido considerar as estatísticas de tabela como parte do código do aplicativo e incluí-las nas definições da tabela.

Por isso, no IRIS hoje você encontra as estatísticas de tabela incorporadas à Definição de Armazenamento. Ao coletar estatísticas de tabela por uma chamada manual para TUNE TABLE ou de maneira implícita pela consulta (veja abaixo), as novas estatísticas são gravadas na Definição de Armazenamento e os planos de consulta existentes para essa tabela são invalidados, para que possam aproveitar as novas estatísticas durante a próxima execução. Por serem parte da Definição de Armazenamento, essas estatísticas farão parte das exportações de classe UDL e, portanto, podem acabar no seu repositório de código-fonte. No caso de estatísticas cuidadosamente verificadas para um aplicativo empacotado, isso é desejado, pois você quer que essas estatísticas específicas levem à geração do plano de consulta para todas as implantações do aplicativo.

A partir de 2021.2, o IRIS coletará automaticamente as estatísticas de tabela no início do planejamento de

consulta ao consultar uma tabela que não possui nenhuma estatística e está qualificada para a amostragem de bloco rápida. Nos nossos testes, os benefícios de trabalhar com estatísticas atualizadas em vez de nenhuma estatística superaram claramente o custo da coleta de estatísticas em tempo real. Para alguns clientes, no entanto, isso teve o lamentável efeito colateral das estatísticas coletadas automaticamente na instância do desenvolvedor terminarem na Definição de Armazenamento no sistema de controle de fonte e, por fim, no aplicativo empacotado. Obviamente, os dados nesse ambiente de desenvolvedor e, portanto, as estatísticas nele talvez não sejam representativos para uma implantação real do cliente e levem a planos de consulta abaixo do ideal.

É fácil evitar essa situação. As estatísticas de tabela podem ser excluídas da exportação da definição de classe usando o qualificador /exportselectivity=0 ao chamar $SYSTEM.OBJ.Export(). O padrão do sistema para essa sinalização pode ser configurado usando $SYSTEM.OBJ.SetQualifiers("/exportselectivity=0"). Depois, deixe para a coleta automática na eventual implantação coletar estatísticas representativas, tornar a coleta de estatísticas explícitas parte do processo de implantação, o que substituirá qualquer coisa que possa ter sido empacotada com o aplicativo, ou gerenciar suas estatísticas de tabela separadamente pelas próprias funções de importação/exportação: $SYSTEM.SQL.Stats.Table.Export() e Import().

A longo prazo, pretendemos mover as estatísticas de tabela para ficar com os dados, em vez de fazer parte do código, e diferenciar mais claramente entre quaisquer estatísticas configuradas explicitamente por um desenvolvedor e as coletadas de dados reais. Além disso, estamos planejando mais automação em relação à atualização periódica dessas estatísticas, com base em quanto os dados da tabela mudam ao longo do tempo.

Conclusão

Neste artigo, descrevemos a função de uma Definição de Armazenamento no mecanismo ObjectRelational do IRIS, como ela é compatível com a evolução do esquema e o que significa incluí-la no seu sistema de controle de fonte. Também descrevemos por que as estatísticas de tabela são atualmente armazenadas nessa Definição de Armazenamento e sugerimos práticas de desenvolvimento para garantir que as implantações de aplicativos acabem com estatísticas representativas dos dados reais do cliente. Conforme mencionado anteriormente, planejamos aprimorar ainda mais esses recursos. Portanto, aguardamos seu feedback sobre a funcionalidade atual e planejada para refinar nosso design conforme apropriado.

#Controle de Fonte #Entrega Contínua #Integração Contínua #SQL #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/cicd-com-sql-do-iris