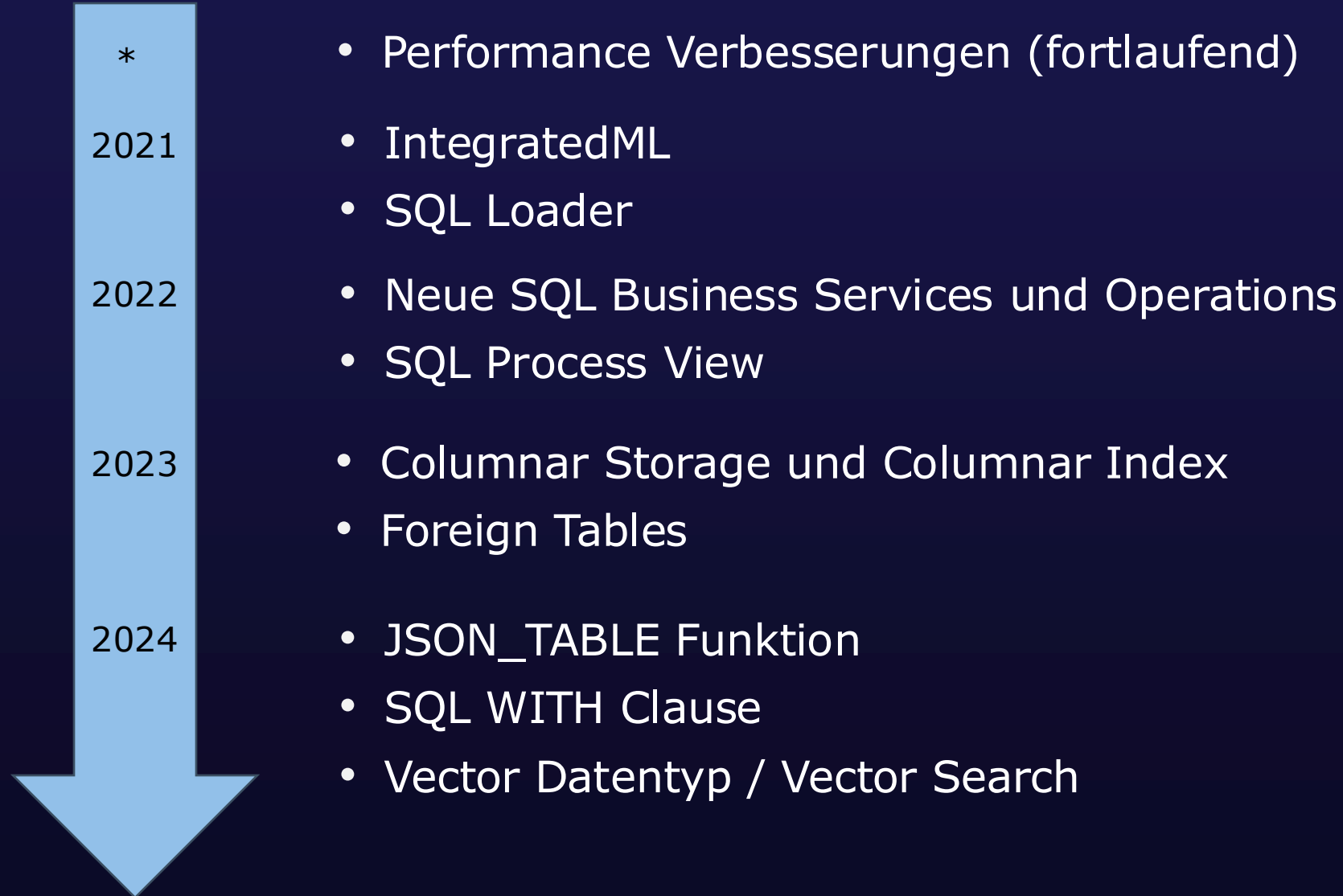


Neue Features in InterSystems IRIS SQL

Stephan Mohr
Sales Engineer



Neue SQL-Features - Überblick



InterSystems IntegratedML – die SQL-Schnittstelle



```
CREATE MODEL <model name> PREDICTING(<label>) FROM <table/view>
```

```
TRAIN MODEL <model name>
```

```
VALIDATE MODEL <model name> FROM <table/view>
```

```
SELECT PREDICT(<model name>) FROM <table/view/record>
```

```
SELECT PROBABILITY(<model name> FOR <label value>)  
FROM <table/view/record>
```

Neu in 2023.3

```
CREATE TIME SERIES MODEL <model name> PREDICTING(<label1, label2, ...>)  
BY (<timestep>) FROM <table/view>
```

```
SELECT WITH PREDICTIONS(<model name>) <columns> FROM <table/view>
```

SQL Loader



- Daten von einer CSV-Datei oder einer JDBC-Quelle effizient laden
- Ziel-Tabelle muss zuvor definiert werden
- Daten werden hinzugefügt und nicht überschrieben
- Laden von Dateien

```
LOAD DATA FROM FILE <file path> INTO <table name>
```

- Laden von JDBC Quelle

```
LOAD DATA FROM JDBC CONNECTION <jdbcConnection> TABLE <jdbcTable>  
INTO <table name>
```

- Siehe Dokumentation für verschiedene Varianten mit Angaben der Spalten und Datentypen sowie die LOAD BULK DATA Option

Low-Code SQL mit InterSystems IRIS Interoperabilität



- Neue Standard Business Services und Operations
 - `EnsLib.SQL.Service.GenericService`
 - `EnsLib.SQL.Service.ProcService`
 - `EnsLib.SQL.Operation.GenericOperation`
 - `EnsLib.SQL.Operation.ProcOperation`
- Konfigurierbare JDBC oder ODBC Datenquellen
- Keine eigenen Message-Klassen notwendig
- SQL-Daten werden standardmäßig in ein Dynamic Object geschrieben und stehen als JSON zur Verfügung

The screenshot shows the configuration window for `EnsLib.SQL.Service.GenericService`. It features a tabbed interface with 'Einstellungen' (Settings) selected. The settings include:

- Übernehmen** (Take over) button and a search field.
- Aufrufintervall** (Call interval) set to 5.
- Externe Registrierungs-ID** (External registration ID) dropdown.
- DSN** (Data Source Name) dropdown.
- Anmeldeinformationen** (Login information) dropdown with a search icon.
- Zielkonfigurationsnamen** (Target configuration names) dropdown.
- Data** section with:
 - Abfrage** (Query) text area.
 - Parameter** (Parameter) text area.
 - SQL-Datentypen der Parameter** (SQL data types of parameters) text area.
 - Abfrage löschen** (Delete query) button.
 - Schlüsselfeldname** (Key field name) text area with 'ID' entered.

The screenshot shows the configuration window for `EnsLib.SQL.Operation.GenericOperation`. It features a tabbed interface with 'Einstellungen' (Settings) selected. The settings include:

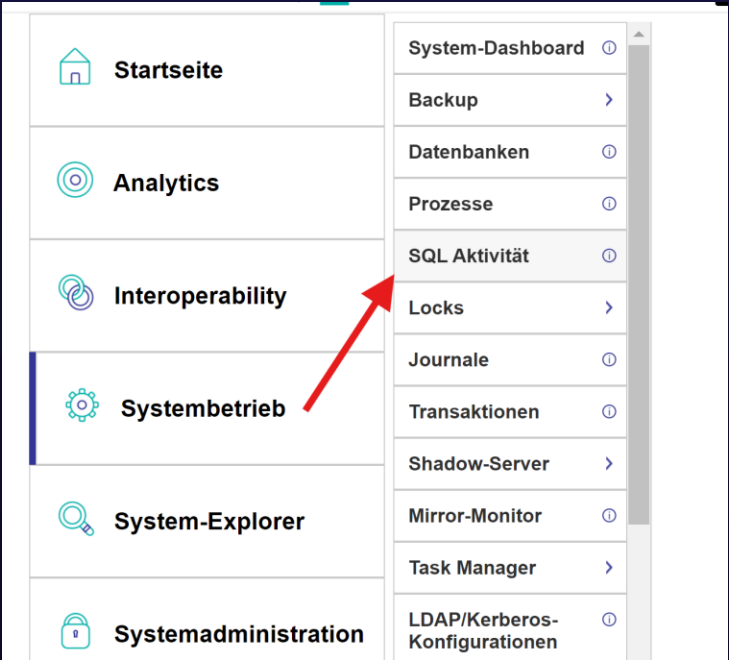
- Übernehmen** (Take over) button and a search field.
- Externe Registrierungs-ID** (External registration ID) dropdown.
- DSN** (Data Source Name) dropdown.
- Anmeldeinformationen** (Login information) dropdown with a search icon.
- Verbindungseinstellungen** (Connection settings) section.
- Data** section with:
 - Abfrage** (Query) text area.
 - Eingabeparameter** (Input parameters) section with a 'Hinzufügen' (Add) button.
 - SQL-Datentypen der Parameter** (SQL data types of parameters) text area.
 - RequestClass** dropdown.
 - Antwortklasse** (Response class) dropdown.
 - Kürzen zulassen** (Allow truncation) checkbox.
- Zusätzliche Einstellungen** (Additional settings) section.

SQL Process View



- Auflistung der aktuell laufenden SQL-Anweisungen
- Abrufbar über SQL oder das System Management Portal
- Lang laufende Abfragen ausfindig machen

```
SELECT * FROM INFORMATION_SCHEMA.CURRENT_STATEMENTS
```



System > Aktuelle SQL-Anweisungen

Aktuelle SQL-Anweisungen

Letzte Aktualisierung: 2024-08-29 16:01:25.87

Derzeit werden SQL-Anweisungen ausgeführt

The following is a list of actively running SQL statements for this instance:

Filter:

Seitenformat: 0

Max. Zeilenanzahl: 1000

Ergebnisse: 1

Seite: 1 von 1

Prozess	Benutzer	NameSpace	Typ der Abfrageausführung	Verstrichene Zeit	Anweisung ID	Anweisung
» 23076	_SYSTEM	%SYS	DynamicQuery	0.004	374	SELECT (SERVER ((' ')) PROCE...

Execution statistics

	Overall	Last week
Times executed	16	2
Average runtime	0.0048 sec	0.0055 sec
Runtime standard deviation	0.0012	0.0015
Average rowcount	1 row(s)	1 row(s)
Average commands	2,130 commands	2,206 commands

Ausgewählte Kontoauszugsdetails

Die Ausführung dieser Erklärung wurde bereits beendet

Process	
SQL Statement ID	374
User	_SYSTEM
Transaction?	Nein
Start/Execution time	2024-08-29 16:01:25 (Ausführung beendet)
Statement	DECLARE C CURSOR FOR SELECT (SERVER ((' ')) PROCESSID ((' ')) STATEMENTINDEXHASH) AS CURRTMTID , SERVER , PROCESSID , USERNAME , NAMESPACE , CASE WHEN PARENT IS NULL THEN QUERYRUNTYPE ELSE (PARENTTYPE ((' Query'))) END AS QUERYRUNTYPEEXEC , QUERYRUNTYPE , ROUND (EXECUTIONDURATION , 3) AS EXECUTIONDURATION , SQLSTATEMENTID , STATEMENTINDEXHASH , TP_NESTINGLEVEL AS TP_NESTINGLEVEL , PARAMETERS , CACHEDQUERY , CALLERNAME , CURRENTWORKERCOUNT , EXECUTIONSTART AS STARTTIME , EXECUTIONSTARTUTC AS STARTTIMEUTC , PARENTTYPE , PARENT , CHILDSTATEMENTS FROM INFORMATION_SCHEMA . CURRENT_STATEMENTS ORDER BY STATEMENTORDER

plan & details

Mehr SQL Performance durch

Indizes



- Erheblicher Performance-Gewinn beim Suchen, Filtern, Sortieren und bei JOIN-Operationen
- Steigerung der Effizienz (weniger Global-Zugriffe und auszuführende Codezeilen)
- (!) Zusätzlicher Bedarf an Speicherplatz
- (!) Erhöhte Anzahl von Operationen bei INSERT, UPDATE, DELETE
- Verschiedene Index-Typen je nach Anwendungsfall

Typ	Anwendung
Standard	Für die meisten Anwendungsfälle
Bitmap	Spalten mit wenigen unterschiedlichen Werten
Bitslice	Numerische Werte für SUM, COUNT, AVG Abfragen

Neu: Columnar Index



- Einsatz in Tabellen mit dem klassischen zeilenbasiertem Speicherlayout
- Speichert eine Kopie der Daten eines bestimmten Feldes in einem komprimierten Vektorformat
- Geeignet für numerische Werte und Strings mit geringer Kardinalität
- Hohe Performance und signifikant weniger Global-Referenzen bei Aggregationen wie SUM oder AVG und bei RANGE Bedingungen
- Definition mit SQL

```
CREATE COLUMNAR INDEX <index name> ON <table>(<column>)
```

- Definition in einer Klasse

```
Index <index name> On <property> [ Type = columnar ]
```


Columnar Storage



- Freie Wahl des Speicher-Layouts für Daten einer Klasse/Tabelle (Global Ebene)
- Speicherung als Zeilen (Rows), Spalten (Columns) oder gemischt

Row based storage

	ID	Name	Age	Address	City	State
1	49203921	SMITH	88	899 FIRST ST	JUNO	AL
2	35004920	CHIN	37	16137 MAIN ST	POMONA	CA
3	10464748	HANDU	12	42 JUNE ST	CHICAGO	IL

49203921, SMITH, 88, 899 FIRST ST, ...

35004920, CHIN, 37, 16137 MAIN ST, ...

10464748, HANDU, 12, 42 JUNE ST, ...

Row 1

Row 2

Row 3

Columnar Storage



Columnar based storage

ID	Name	Age	Address	City	State
49203921	SMITH	88	899 FIRST ST	JUNO	AL
35004920	CHIN	37	16137 MAIN ST	POMONA	CA
10464748	HANDU	12	42 JUNE ST	CHICAGO	IL
...					

49203921, 35004920, 10464748, ...

SMITH, CHIN, HANDU, ...

88, 37, 12, ...

Column ID

Column Name

Column Age

etc.

Columnar Storage



Row & Columnar hybrid used storage

	ID	Name	Address	City	Age	State
1	49203921	SMITH	899 FIRST ST	JUNO	88	AL
2	35004920	CHIN	16137 MAIN ST	POMONA	37	CA
3	10464748	HANDU	42 JUNE ST	CHICAGO	12	IL

49203921, SMITH, 899 FIRST ST, ...	35004920, CHIN, 16137 MAIN ST, ...	10464748, HANDU, 42 JUNE ST, ...
------------------------------------	------------------------------------	----------------------------------

Row 1

Row 2

Row 3

88, 37, 12, ...

AL, CA, IL, ...

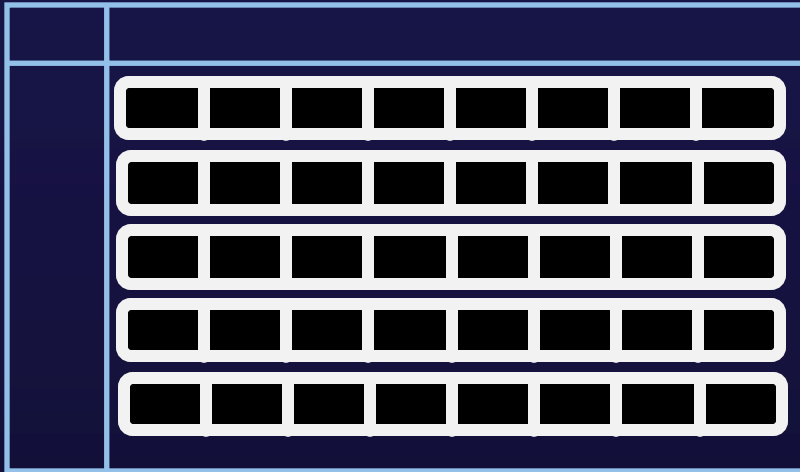
Column Age

Column State

Row Storage vs. Columnar Storage

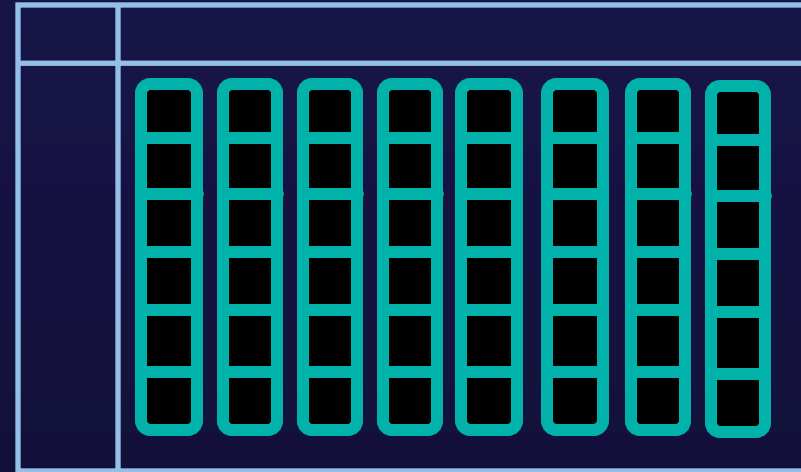


Row Storage



- Schnelle Transaktionen – row inserts, updates & deletes
- Fokus auf Latenz
- Daten zeilenweise zusammenhalten, basierend auf ihrer Erstellung
- Zeilenzentriertes I/O – Abruf ganzer Zeile

Columnar Storage



- Schnelle Aggregationen – komplexe Abfragen, vektorisierte Verarbeitung
- Fokus auf Durchsatz
- Daten spaltenweise zusammenhalten basierend auf den Lesevorgang
- Selectives I/O – Aggregationen liefern, nur benötigte Daten lesen

Storage Type definieren



- Definition mit SQL

```
CREATE TABLE .... WITH STORAGETYPE = ROW
```

```
CREATE TABLE .... WITH STORAGETYPE = COLUMNAR
```

Hybrid `<column name> <column type> WITH STORAGETYPE = COLUMNAR`

- Definition in einer Klasse

```
Parameter STORAGEDEFAULT = "row";
```

```
Parameter STORAGEDEFAULT = "columnar";
```

Hybrid `Property <property> As <data type>(STORAGEDEFAULT = "columnar");`

Foreign Tables



- Projektion von Daten aus externen Datenquellen
- Erlaubt Abfragen in Kombination mit nativen Daten von InterSystems IRIS
- Keine komplexe Datenmigration oder Datenduplizierung notwendig
- Einbindung von Datenbanken über JDBC oder Dateien (CSV-Format)
- Schritt 1: Definition eines Foreign Servers

```
CREATE FOREIGN SERVER <server name>  
FOREIGN DATA WRAPPER CSV HOST '<directory>'
```

```
CREATE FOREIGN SERVER <server name>  
FOREIGN DATA WRAPPER JDBC CONNECTION '<connection name>'
```

Foreign Tables



- Schritt 2: Definition einer Foreign Table

```
CREATE FOREIGN TABLE <table name> (  
    <column name> <column type>,  
    <column name> <column type>,  
    ...  
)  
SERVER <server name> FILE '<file name>'
```

```
CREATE FOREIGN TABLE <table name> (  
    <column name> <columns type>,  
    <column name> <columns type>,  
    ...  
)  
SERVER <server name> TABLE '<table name>'
```

Foreign Table Demo mit CSV Dateizugriff



```
CREATE FOREIGN SERVER Sample.TestFile
  FOREIGN DATA WRAPPER CSV HOST 'C:\Test\Excel'
```

```
CREATE FOREIGN TABLE Sample.Employees (
  EEID VARCHAR(6),
  FullName VARCHAR(30),
  JobTitle VARCHAR(30),
  Department VARCHAR(30),
  BusinessUnit VARCHAR(30),
  Gender VARCHAR(30),
  Ethnicity VARCHAR(30),
  Age VARCHAR(30),
  HireDate VARCHAR(30),
  AnnualSalary VARCHAR(30),
  Bonus VARCHAR(30),
  Country VARCHAR(30),
  City VARCHAR(30),
  ExitDate VARCHAR(30)
```

```
)
SERVER Sample.TestFile FILE 'Employees.csv'
```

```
SELECT * FROM Sample.Employees
```

```
SELECT * FROM Sample.Employees WHERE Department = 'IT'
ORDER BY Age
```

```
DROP FOREIGN SERVER Sample.TestFile
CASCADE
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	E02387	Emily Davis	Sr. Manger	IT,Research & Development	Female	Black	55	08.04.2016	\$141604	15%	United States	Seattle	16.10.2021
2	E04105	Theodore Dinh	Technical Architect	IT,Manufacturing	Male	Asian	59	29.11.1997	\$99975	0%	China	Chongqing	
3	E02572	Luna Sanders	Director	Finance,Speciality Products	Female	Caucasian	50	26.10.2006	\$163099	20%	United States	Chicago	
4	E02832	Penelope Jordan	Computer Systems Manager	IT,Manufacturing	Female	Caucasian	26	27.09.2019	\$84913	7%	United States	Chicago	
5	E01639	Austin Vo	Sr. Analyst	Finance,Manufacturing	Male	Asian	55	20.11.1995	\$95409	0%	United States	Phoenix	
6	E00644	Joshua Gupta	Account Representative	Sales,Corporate	Male	Asian	57	24.01.2017	\$50994	0%	China	Chongqing	
7	E01550	Ruby Barnes	Manager	IT,Corporate	Female	Caucasian	27	01.07.2020	\$119746	10%	United States	Phoenix	
8	E04332	Luke Martin	Analyst	Finance,Manufacturing	Male	Black	25	16.05.2020	\$41336	0%	United States	Miami	20.05.2021
9	E04533	Easton Bailey	Manager	Accounting,Manufacturing	Male	Caucasian	29	25.01.2019	\$113527	6%	United States	Austin	
10	E03838	Madeline Walker	Sr. Analyst	Finance,Speciality Products	Female	Caucasian	34	13.06.2018	\$77203	0%	United States	Chicago	
11	E00591	Savannah Ali	Sr. Manger	Human Resources,Manufacturing	Female	Asian	36	11.02.2009	\$157333	15%	United States	Miami	
12	E03344	Camila Rogers	Controls Engineer	Engineering,Speciality Products	Female	Caucasian	27	21.10.2021	\$109851	0%	United States	Seattle	
13	E00530	Eli Jones	Manager	Human Resources,Manufacturing	Male	Caucasian	59	14.03.1999	\$105086	9%	United States	Austin	
14	E04239	Everleigh Ng	Sr. Manger	Finance,Research & Development	Female	Asian	51	10.06.2021	\$146742	10%	China	Shanghai	
15	E03496	Robert Yang	Sr. Analyst	Accounting,Speciality Products	Male	Asian	31	04.11.2017	\$97078	0%	United States	Austin	09.03.2020
16	E00549	Isabella Xi	Vice President	Marketing,Research & Development	Female	Asian	41	13.03.2013	\$249270	30%	United States	Seattle	
17	E00163	Bella Powell	Director	Finance,Research & Development	Female	Black	65	04.03.2002	\$175837	20%	United States	Phoenix	
18	E00884	Camila Silva	Sr. Manger	Marketing,Speciality Products	Female	Latino	64	01.12.2003	\$154828	13%	United States	Seattle	
19	E04116	David Barnes	Director	IT,Corporate	Male	Caucasian	64	03.11.2013	\$186503	24%	United States	Columbus	
20	E04625	Adam Dang	Director	Sales,Research & Development	Male	Asian	45	09.07.2002	\$166331	18%	China	Chongqing	

EEID	FullName	JobTitle	Department	BusinessUnit	Gender	Ethnicity	Age	HireDate	AnnualSalary	Bonus	Country	City	ExitDate
E02387	Emily Davis	Sr. Manger	IT	Research & Development	Female	Black	55	08.04.2016	\$141604	15%	United States	Seattle	16.10.2021
E04105	Theodore Dinh	Technical Architect	IT	Manufacturing	Male	Asian	59	29.11.1997	\$99975	0%	China	Chongqing	
E02572	Luna Sanders	Director	Finance	Speciality Products	Female	Caucasian	50	26.10.2006	\$163099	20%	United States	Chicago	
E02832	Penelope Jordan	Computer Systems Manager	IT	Manufacturing	Female	Caucasian	26	27.09.2019	\$84913	7%	United States	Chicago	
E01639	Austin Vo	Sr. Analyst	Finance	Manufacturing	Male	Asian	55	20.11.1995	\$95409	0%	United States	Phoenix	
E00644	Joshua Gupta	Account Representative	Sales	Corporate	Male	Asian	57	24.01.2017	\$50994	0%	China	Chongqing	
E01550	Ruby Barnes	Manager	IT	Corporate	Female	Caucasian	27	01.07.2020	\$119746	10%	United States	Phoenix	
E04332	Luke Martin	Analyst	Finance	Manufacturing	Male	Black	25	16.05.2020	\$41336	0%	United States	Miami	20.05.2021
E04533	Easton Bailey	Manager	Accounting	Manufacturing	Male	Caucasian	29	25.01.2019	\$113527	6%	United States	Austin	
E03838	Madeline Walker	Sr. Analyst	Finance	Speciality Products	Female	Caucasian	34	13.06.2018	\$77203	0%	United States	Chicago	
E00591	Savannah Ali	Sr. Manger	Human Resources	Manufacturing	Female	Asian	36	11.02.2009	\$157333	15%	United States	Miami	
E03344	Camila Rogers	Controls Engineer	Engineering	Speciality Products	Female	Caucasian	27	21.10.2021	\$109851	0%	United States	Seattle	
E00530	Eli Jones	Manager	Human Resources	Manufacturing	Male	Caucasian	59	14.03.1999	\$105086	9%	United States	Austin	
E04239	Everleigh Ng	Sr. Manger	Finance	Research & Development	Female	Asian	51	10.06.2021	\$146742	10%	China	Shanghai	
E03496	Robert Yang	Sr. Analyst	Accounting	Speciality Products	Male	Asian	31	04.11.2017	\$97078	0%	United States	Austin	09.03.2020
E00549	Isabella Xi	Vice President	Marketing	Research & Development	Female	Asian	41	13.03.2013	\$249270	30%	United States	Seattle	
E00163	Bella Powell	Director	Finance	Research & Development	Female	Black	65	04.03.2002	\$175837	20%	United States	Phoenix	
E00884	Camila Silva	Sr. Manger	Marketing	Speciality Products	Female	Latino	64	01.12.2003	\$154828	13%	United States	Seattle	
E04116	David Barnes	Director	IT	Corporate	Male	Caucasian	64	03.11.2013	\$186503	24%	United States	Columbus	
E04625	Adam Dang	Director	Sales	Research & Development	Male	Asian	45	09.07.2002	\$166331	18%	China	Chongqing	
E03680	Elias Alvarado	Sr. Manger	IT	Manufacturing	Male	Latino	56	09.01.2012	\$146140	10%	Brazil	Manaus	
E04732	Eva Rivera	Director	Sales	Manufacturing	Female	Latino	36	02.04.2021	\$151703	21%	United States	Miami	
E03484	Logan Rivera	Director	IT	Research & Development	Male	Latino	59	24.05.2002	\$172787	28%	Brazil	Rio de Janeiro	
E00671	Leonardo Dixon	Analyst	Sales	Speciality Products	Male	Caucasian	37	05.09.2019	\$49998	0%	United States	Seattle	
E02071	Mateo Her	Vice President	Sales	Speciality Products	Male	Asian	44	02.03.2014	\$207172	31%	China	Chongqing	

JSON_TABLE



- Funktion zur Abfrage von JSON-Daten mit SQL (SELECT ... FROM)
- Umwandlung von JSON-Werten in ein relationales Format (Rückgabe einer Tabelle)
- Beschreibung von einfachen und verschachtelten JSON-Strukturen durch SQL/JSON-Pfadausdrücke (path language expressions)
- Auswahl der Spalten über ein Mapping
- Unterstützung von SQL-Funktionen, Aggregationen, Filtern und Joins
- Syntax:

```
JSON_TABLE( <json_value>, <json-path> <col-mapping>)
```

- Der JSON-Wert kann ein literaler String, eine Funktion, die einen JSON-String zurückliefert (z.B. eine REST API) oder eine Spalte mit einem JSON-Wert sein

JSON_TABLE



- Beispiele:

```
SELECT my_value FROM JSON_TABLE(  
    '[{"number":"two"}, {"number":"three"}, {"number":"four"}]',  
    '$'  
    COLUMNS ( my_value varchar(20) PATH '$.number' )  
)
```

```
SELECT TOP 10 name, username, email  
FROM Sample."User", JSON_TABLE (  
    Sample."User".JSONUserContent,  
    '$'  
    COLUMNS (  
        name VARCHAR(100) PATH '$.name',  
        username VARCHAR(100) PATH '$.username',  
        email VARCHAR(100) PATH '$.email'  
    )  
) as jt  
ORDER BY name, username
```

SQL-WITH



- Ermöglicht die einfache Verwendung einer Unterabfrage innerhalb einer Hauptabfrage
- Die Hauptabfrage muss ein SELECT-Ausdruck sein
- Verbesserte Lesbarkeit und mehrfache Referenzierung möglich
- Beispiel: Durchschnittliche Bestellmenge pro Produkt

Verschachteltes SQL

```
SELECT AVG(Total) AS  
    average_product_quantity  
FROM  
    (SELECT SUM(Quantity) AS Total  
    FROM OrderDetails  
    GROUP BY ProductID)
```

SQL-WITH

```
WITH cte_quantity AS  
    (SELECT SUM(Quantity) AS Total  
    FROM OrderDetails  
    GROUP BY ProductID)  
  
SELECT AVG(Total) AS  
    average_product_quantity  
FROM cte_quantity
```

Datentyp VECTOR



- Dedizierter Datentyp zur Speicherung von Vektor-Daten in einem komprimierten und performanten Format
- Nutzt Optimierungen auf Chip-Ebene zur Ausführung von Vektor-Operationen
- Jedes Element hat den gleichen Datentyp: Integer, Double, Decimal oder String
- Dient als Grundlage für Columnar Storage und Vector Search für KI-Anwendungen
- Definition mit SQL

```
CREATE TABLE <table name> (<column> VECTOR(<type>,<length>), ...
```

- Definition in einer Klasse

```
Property <name> As %Vector(DATATYPE = <datatype>, LEN = <length>)
```

VECTOR Funktionen (SQL)



- TO_VECTOR
 - Konvertiert Eingabedaten, z.B. einen String, in einen Vector des angegebenen Typs
 - Kann in INSERT Kommandos verwendet werden, um eine Vektor-Spalte zu füllen
- Syntax:

```
TO_VECTOR(<daten>, <type>, <length>)
```

- Beispiel:

```
INSERT INTO Words.Embeddings (word,embedding)  
VALUES ('person', TO_VECTOR('1,3,5,7,9',int))
```



VECTOR Funktionen (SQL)

- VECTOR_COSINE
 - Gibt die Kosinusähnlichkeit zwischen zwei Vektoren zurück
 - Beide Vektoren müssen einen numerischen Typ gleicher Länge haben
 - Das Ergebnis vom Typ Double liegt zwischen -1 und 1, wobei der höchste Wert die höchste Übereinstimmung bedeutet (1 = identisch)
- Syntax:

```
VECTOR_COSINE(<vec1>, <vec2>)
```

- Beispiel:

```
SELECT VECTOR_COSINE(  
  TO_VECTOR('6,4,5',integer), TO_VECTOR('1,4,3',integer))
```

```
0,8269317890951972686
```



VECTOR Funktionen (SQL)

- VECTOR_DOT_PRODUCT
 - Berechnet das Skalarprodukt zwischen zwei Vektoren
 - Beide Vektoren müssen einen numerischen Typ gleicher Länge haben
 - Bevorzuge Funktion bei Einheitsvektoren
- Syntax:

```
VECTOR_DOT_PRODUCT(<vec1>, <vec2>)
```

- Beispiel:

```
SELECT VECTOR_DOT_PRODUCT(  
TO_VECTOR('6,4,5',integer), TO_VECTOR('1,4,3',integer))
```

Vector Search



- Unstrukturierte Daten können durch ein Embedding Modell (KI) in eine Struktur, einen Vektor, transformiert werden
- Vektoren können die semantische Bedeutung von Inhalten repräsentieren
- Wörter oder Sätze mappen Wörter oder Sätze in einen geometrischen Raum mit hunderten von Dimensionen, wobei semantische Ähnlichkeiten nahe beieinander liegen
- Embeddings können in InterSystems IRIS in den komprimierten und leistungsfähigen Vektor-Datentyp gespeichert werden
- Semantische Ähnlichkeiten von Eingabedaten zu vorhandenen Daten können durch einfache SQL-Funktionen auf den Vektoren ermittelt werden.
- Anwendung:

```
SELECT TOP 5 <text column> FROM <table> ...  
ORDER BY VECTOR_DOT_PRODUCT(<vector column>, <input vector>) DESC
```




**Vielen
Dank!**