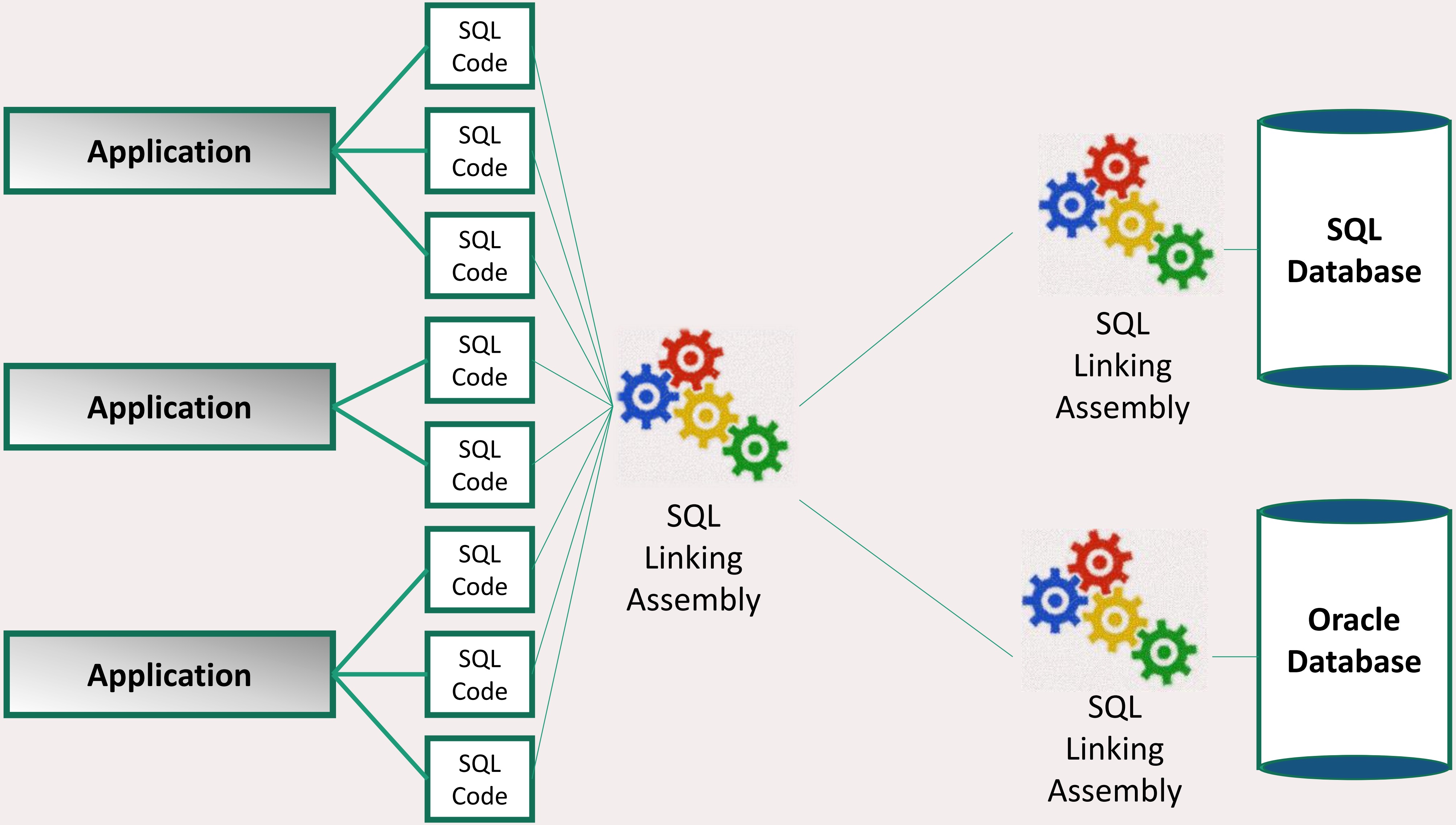
The Next Advance in SQL Generation Technology

Vizier Finder™ Overview

- Vizier is a next generation SQL Code Generator that greatly reduces application coding requirements, saving time and resources.
- It is available for .net and Java development environments.
- It an be used to generate web, desktop and client server applications.
- As an API, it can be used to build new applications, or to enhance existing ones.
- Vizier speeds up application processing, manages inexact string processing, and reduces software lifecycle maintenance costs.

Current Development Paradigm



Application

Application

Application

SQL Code

SQL Code

SQL Code

SQL Code

SQL Code

SQL Code

SQL Code

SQL Code

SQL Linking Assembly

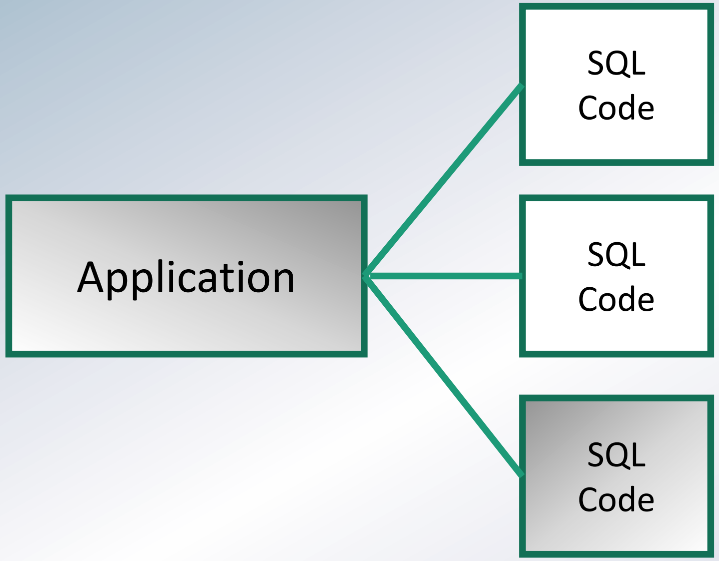
SQL Linking Assembly

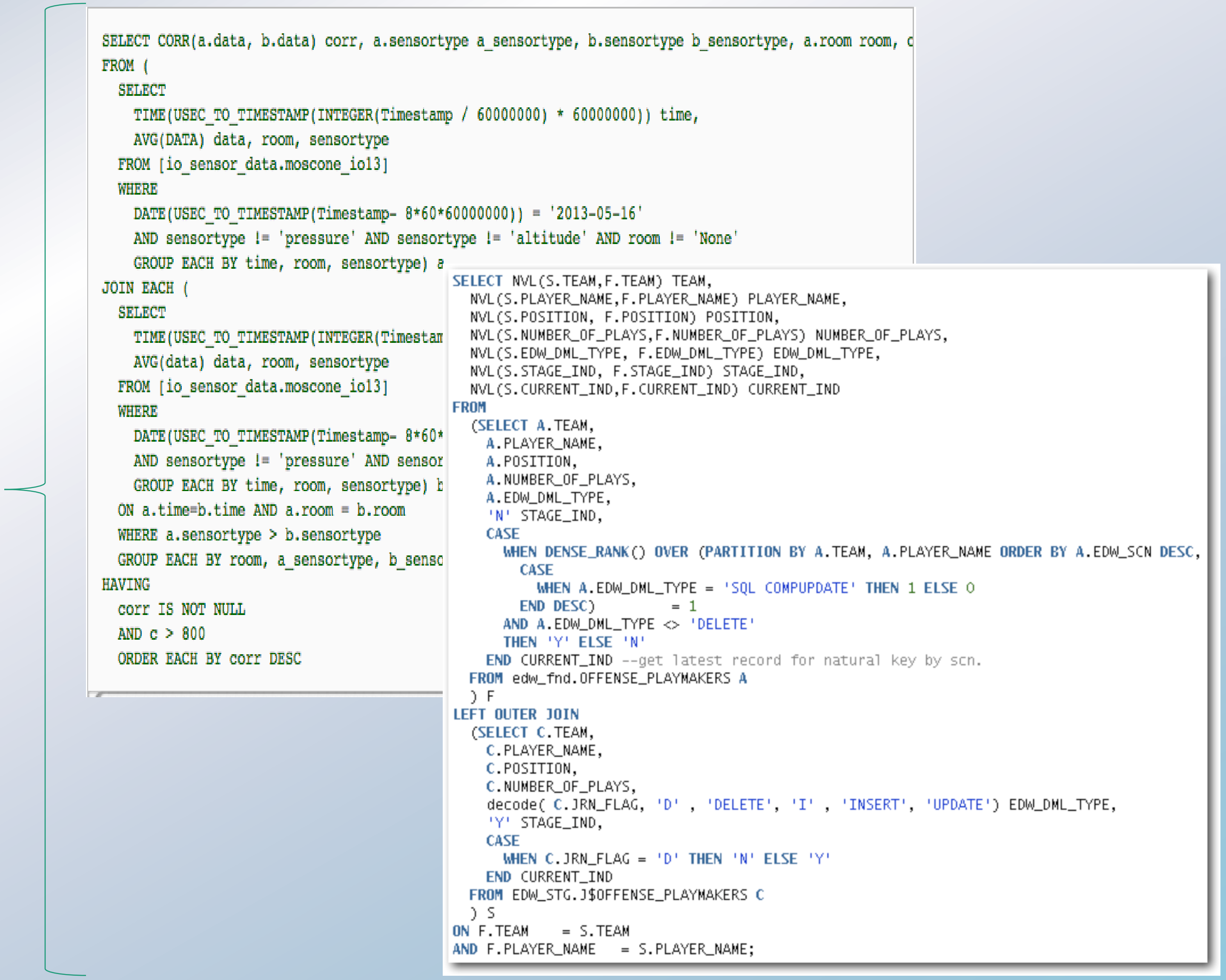
SQL Linking Assembly

SQL Database

Oracle Database

SQL Code Generation Sample





```
SELECT CORR(a.data, b.data) corr, a.sensortype a_sensortype, b.sensortype b_sensortype, a.room room, c
FROM (
  SELECT
    TIME(USEC_TO_TIMESTAMP(INTEGER(Timestamp / 60000000) * 60000000)) time,
    AVG(DATA) data, room, sensortype
  FROM [io_sensor_data.moscone_io13]
  WHERE
    DATE(USEC_TO_TIMESTAMP(Timestamp- 8*60*60000000)) = '2013-05-16'
    AND sensortype != 'pressure' AND sensortype != 'altitude' AND room != 'None'
    GROUP EACH BY time, room, sensortype) a
JOIN EACH (
  SELECT
    TIME(USEC_TO_TIMESTAMP(INTEGER(Timestam
    AVG(data) data, room, sensortype
  FROM [io_sensor_data.moscone_io13]
  WHERE
    DATE(USEC_TO_TIMESTAMP(Timestamp- 8*60*
    AND sensortype != 'pressure' AND sensor
    GROUP EACH BY time, room, sensortype) b
  ON a.time=b.time AND a.room = b.room
  WHERE a.sensortype > b.sensortype
  GROUP EACH BY room, a_sensortype, b_senso
HAVING
  corr IS NOT NULL
  AND c > 800
  ORDER EACH BY corr DESC
```

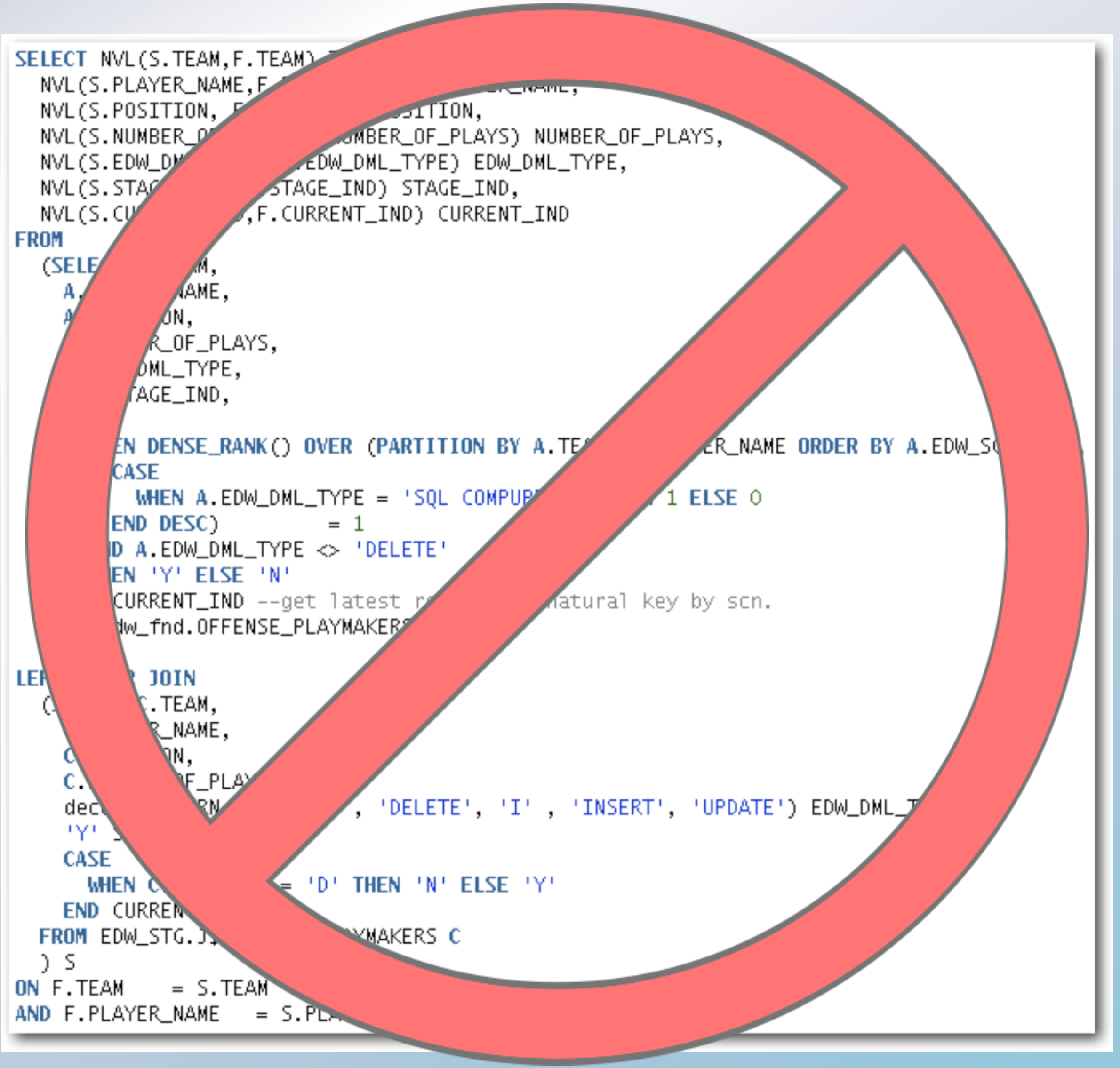
```
SELECT NVL(S.TEAM,F.TEAM) TEAM,
  NVL(S.PLAYER_NAME,F.PLAYER_NAME) PLAYER_NAME,
  NVL(S.POSITION, F.POSITION) POSITION,
  NVL(S.NUMBER_OF_PLAYS,F.NUMBER_OF_PLAYS) NUMBER_OF_PLAYS,
  NVL(S.EDW_DML_TYPE, F.EDW_DML_TYPE) EDW_DML_TYPE,
  NVL(S.STAGE_IND, F.STAGE_IND) STAGE_IND,
  NVL(S.CURRENT_IND,F.CURRENT_IND) CURRENT_IND
FROM
  (SELECT A.TEAM,
    A.PLAYER_NAME,
    A.POSITION,
    A.NUMBER_OF_PLAYS,
    A.EDW_DML_TYPE,
    'N' STAGE_IND,
    CASE
      WHEN DENSE_RANK() OVER (PARTITION BY A.TEAM, A.PLAYER_NAME ORDER BY A.EDW_SCN DESC,
        CASE
          WHEN A.EDW_DML_TYPE = 'SQL COMPUPDATE' THEN 1 ELSE 0
        END DESC)          = 1
      AND A.EDW_DML_TYPE <> 'DELETE'
      THEN 'Y' ELSE 'N'
    END CURRENT_IND --get latest record for natural key by scn.
  FROM edw_fnd.OFFENSE_PLAYMAKERS A
  ) F
LEFT OUTER JOIN
  (SELECT C.TEAM,
    C.PLAYER_NAME,
    C.POSITION,
    C.NUMBER_OF_PLAYS,
    decode( C.JRN_FLAG, 'D' , 'DELETE', 'I' , 'INSERT', 'UPDATE') EDW_DML_TYPE,
    'Y' STAGE_IND,
    CASE
      WHEN C.JRN_FLAG = 'D' THEN 'N' ELSE 'Y'
    END CURRENT_IND
  FROM EDW_STG.J$OFFENSE_PLAYMAKERS C
  ) S
ON F.TEAM    = S.TEAM
AND F.PLAYER_NAME   = S.PLAYER_NAME;
```

Current Development Paradigm

- Each Application Design Change Requires
  - Documentation of Coding Solution
  - Generation of Coding Solution
  - Compiling Coding Solution
  - Building Coding Solution
  - Configuration management of Coding Solution
  - Testing of Code

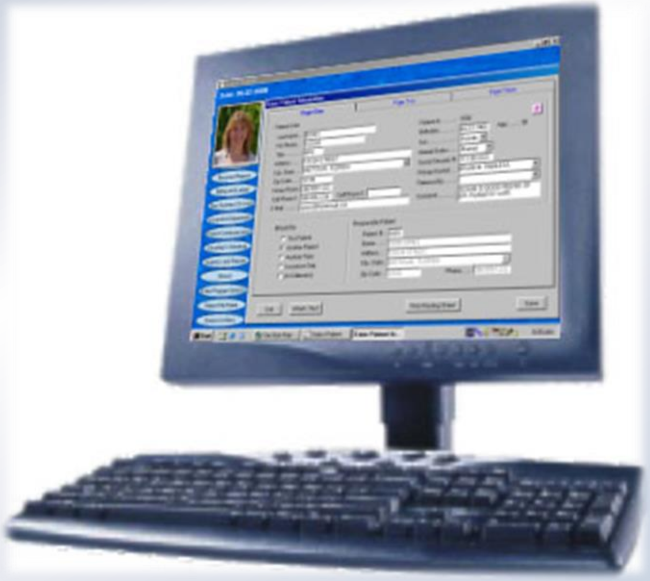
- Repeat above processes for
  - Unsuccessful Tests
  - Changes in Database Design
  - Changes in Application Design

What if you had a way to build applications with a lot less coding?

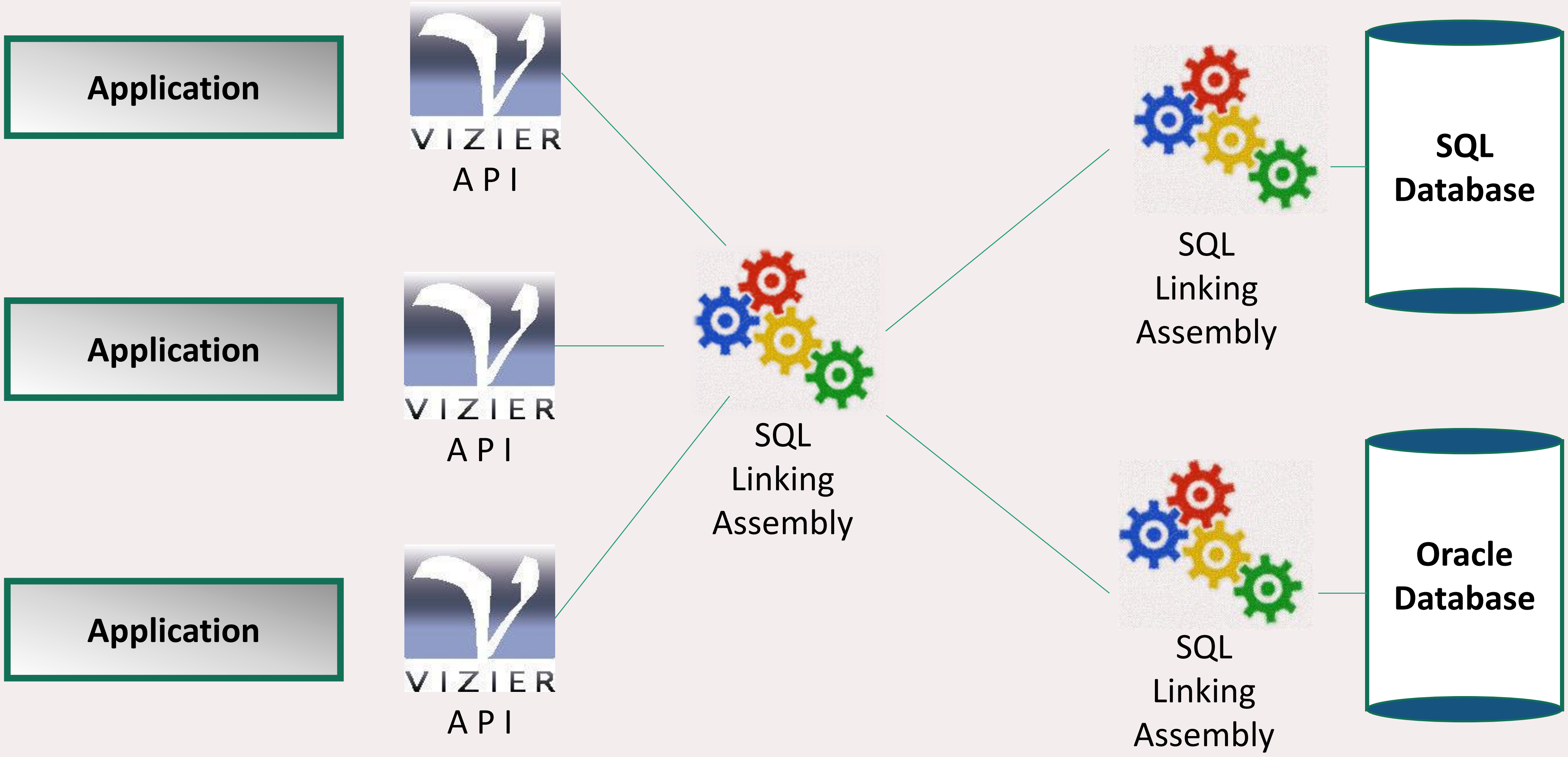


```
SELECT NVL(S.TEAM,F.TEAM)
  NVL(S.PLAYER_NAME,F.                ER_NAME,
  NVL(S.POSITION, F               OSITION,
  NVL(S.NUMBER_O              UMBER_OF_PLAYS) NUMBER_OF_PLAYS,
  NVL(S.EDW_DM              .EDW_DML_TYPE) EDW_DML_TYPE,
  NVL(S.STAG              STAGE_IND) STAGE_IND,
  NVL(S.CU              ,F.CURRENT_IND) CURRENT_IND
FROM
  (SELE         M,
    A.         NAME,
    A          ON,
              R_OF_PLAYS,
              DML_TYPE,
              AGE_IND,

             EN DENSE_RANK() OVER (PARTITION BY A.TE          ER_NAME ORDER BY A.EDW_S
             CASE
               WHEN A.EDW_DML_TYPE = 'SQL COMPUP            1 ELSE 0
             END DESC)          = 1
             D A.EDW_DML_TYPE <> 'DELETE'
             EN 'Y' ELSE 'N'
             CURRENT_IND --get latest r          natural key by scn.
             dw_fnd.OFFENSE_PLAYMAKERS

LEF         R JOIN
  (         C.TEAM,
             R_NAME,
    C        ON,
    C.       F_PLA
    dec      RN           , 'DELETE', 'I' , 'INSERT', 'UPDATE') EDW_DML_T
    'Y'
    CASE
      WHEN C              = 'D' THEN 'N' ELSE 'Y'
    END CURREN
  FROM EDW_STG.J              YMAKERS C
  ) S
ON F.TEAM     = S.TEAM
AND F.PLAYER_NAME   = S.PL
```

What if you could make your applications access data much faster?



New Development Paradigm

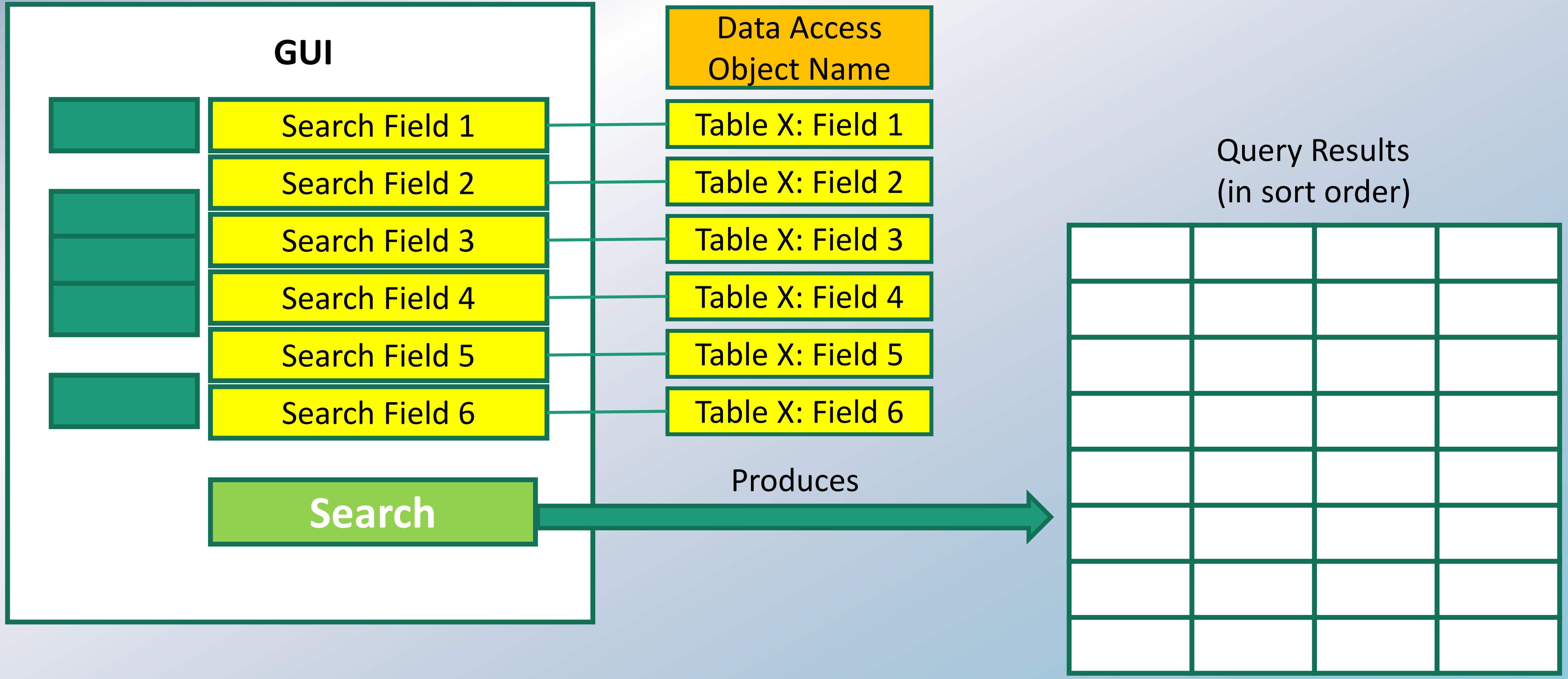


Method of Implementation

You identify the GUI objects

And how you want to link them to your Data Access Object elements

And where you want the results to go



GUI

Search Field 1

Search Field 2

Search Field 3

Search Field 4

Search Field 5

Search Field 6

Search

Data Access Object Name

Table X: Field 1

Table X: Field 2

Table X: Field 3

Table X: Field 4

Table X: Field 5

Table X: Field 6

Produces

Query Results (in sort order)

Local Argument Controls

- For Standard (Structured) Arguments
  - Inexact
  - Exact
  - Not
  - Wildcard
  - Between
  - Less Than or Equal
  - Greater Than or Equal

- For Unstructured Arguments
  - Inexact Phrase
  - Exact Phrase
  - Inexact Word
  - Exact Word

These are controls that are available on a field by field basis and can be set at compile time by the developer, or on run time by the user

The argument controls can be displayed or not, depending on the implementation preference of the developer

Global Argument Controls

- The developer (or user) can also set other controls that can be used to manage search results
  - Fields Weights
  - Results cut-off
  - Search Depth
- For search efforts involving documents, the developer (or user) can manage
  - The number of characters to read in a document
  - The number of document to be processed in a search

Implementation Features

- No scripts to write
- No special system setup required
  - No Indexing
  - No Stored Procedures
  - No Cookies
  - No modifications to the databases
  - No pre-document processing
- No indexes to create or maintain
- Does not write to:
  - The Application GUI
  - Any objects being accessed
- Uses fewer resources than manually written code

Available Environments

- SQL Environments
  - SQL Server
  - Oracle
  - Access
  - DB2
  - No SQL
  - SharePoint
  - Desktops
  - Social Media

- Development Environments
  - .net
  - Java

Goal

To Minimize Life Cycle Costs Associated with
Code Development

Solution

Use Vizier Finder™ Code Assembly to Reduce
Application Coding Requirements,
Saving Valuable Time and Resources

Cylab, Inc.
8500 Leesburg Pike, Suite 405
Vienna, VA 22182

703-379-4818

www.cylabinc.com