# SQL

# נושאים מתקדמים



טלפון: **077-4123211/2** Email: info@iitc.co.il

## Contents


| | |
|---|---|
| 1. Views | 1 |
| 2. Stored Procedures | 8 |
| 3. בדיקות DB | 1 |
| 4. חוברת תרגילים | 1 |
| 5. חוברת פתרונות | 1 |

# Views

IITC - QA
המרכז הישראלי
לטכנולוגיות הבטחת איכות
www.iitc.co.il





## נושאים

- View
- יצירת, שינוי ומחיקת View
- פקודות DML על Views

# View

- View הינו אוביקט במסד הנתונים המאחסן שאילתה הבונה טבלה וירטואלית
- בכל פעם שאנו פונים ל-View הפקודה רצה מחדש מול בסיס הנתונים ואנו רואים טבלה כאילו היתה זו טבלה אמיתית במסד הנתונים
- ה-View כולל נתונים ושדות הנשלפים מטבלאות פיזיות במסד הנתונים




# View
## יתרונות

- פיקוח על המידע המוצג למשתמש
- הסתרת מידע מגורמים שהמידע אינו רלוונטי להם
- הסתרת עיצוב מורכב של בסיס הנתונים
- שיפור ביצועים וזמני תגובה
- מארגן מידע עבור ייצוא לאפליקציות אחרות
- מקל על כתיבת שאילתות מסובכות

# Create View

- יצירת View:

CREATE VIEW view_name [(alias,…)] AS
שאילתה
[WITH CHECK OPTION]

- דוגמא:
- יצירת VIEW המציג את פרטי העובדים שאין להם איזור מוגדר (REGION):

```
CREATE VIEW emp_det_vw AS
SELECT employeeID, lastname, firstname, birthdate
FROM employees
WHERE Region IS NULL
```

- ניתן ליצור VIEW המסתמך על VIEW אחר (קורא נתונים מ-VIEW אחר). שם ה-View עליו מסתמכים יופיע בחלק ה- FROM בהגדרת ה-VIEW החדש.




# הרצת View

- הרצת ה-View ע"י הפקודה:

Select */Column name
From <View_Name>

- דוגמא:

SELECT * FROM emp_det_vw

# View המורכב ממספר טבלאות

- יצירת ה-View ע"י שימוש ב-JOIN
  - דוגמא: יצירת View המציג פרטים לגבי הזמנות שהמשלוח שלהן לא מבוצע בזמן, עבור כל הזמנה כזו יוצג שם הלקוח מספר ההזמנה ותאריך המשלוח

```
CREATE VIEW shipStatusView
AS
SELECT o.OrderId, o.ShippedDate, c.ContactName
FROM Customers c JOIN Orders o
ON c.CustomerId = o.CustomerId
WHERE RequiredDate < ShippedDate
```




# View עם בדיקה

- יצירת View עם אילוץ מסוג Check, המוודא עמידה בתנאי הבדיקה כאשר מכניסים או משנים נתונים
  - דוגמא: יצירת View עם פרטי העובדים שתאריך תחילת העבודה שלהם גדול מתאריך הלידה, עם בדיקה

```
CREATE VIEW emp_det_vw_HireDate
AS
SELECT employeeID, lastname, firstname, birthdate, hiredate
FROM employees
WHERE hiredate > birthdate WITH CHECK OPTION
```

- אם נרצה להכניס (או לעדכן) עובד שתאריך תחילת העבודה שלו יהיה קטן מתאריך לידתו – נקבל שגיאה והעובד לא יכנס לטבלה

# שינוי מבנה ה-View

- שינוי מבנה ה-View ע"י פקודת ALTER:

ALTER VIEW view_name [(alias,…)]
AS
שאילתה

- דוגמא:
- הוספת עמודת Region ל-View: emp_det_vw_HireDate

ALTER VIEW view emp_det_vw_HireDate
AS
SELECT employeeID, lastname, firstname, birthdate,HireDate, Region
FROM employees
WHERE HireDate>BirthDate **WITH CHECK OPTION**




# מחיקת View

- מחיקת View ע"י פקודת Drop:

DROP VIEW view_name

– דוגמא:
מחיקת emp_det_vw_HireDate view

DROP VIEW emp_det_vw_HireDate

# פקודות DML על View

- ניתן להפעיל פקודות DML על View כמו על טבלה רגילה רק בתנאי ש:
  - ה-View אינו Read-Only
  - ה- VIEW מורכב מנתונים של טבלה אחת בלבד
  - אין שימוש במילה Distinct
  - אין שימוש ב-Top כאשר יש שימוש באופציה WITH CHECK
  - אין משפט Group By או Having
  - ה-View אינו כולל פונקציות צבירה
- פקודות ה-DML ירוצו על הטבלה המקורית ממנה בנוי ה-View




# INSERT INTO View

- פקודת INSERT על View:

```
INSERT INTO <View Name>
 VALUES ('value1','value2',…)
```

- דוגמא:

הוספת עובד ל-View: emp_det_vw

```
INSERT INTO emp_det_vw
VALUES ('Floyd', 'Pink', NULL)
```

העובד החדש יתווסף גם ל-View וגם לטבלה המקורית Employees:

| employeeID | lastname | firstname | birthdate |
|---|---|---|---|
| 5 | Buchanan | Steven | 1955-03-04 00:00:00.000 |
| 6 | Suyama | Michael | 1963-07-02 00:00:00.000 |
| 7 | King | Robert | 1960-05-29 00:00:00.000 |
| 9 | Dodsworth | Anne | 1966-01-27 00:00:00.000 |
| 10 | Floyd | Pink | NULL |

# UPDATE VIEW

- פקודת Update על View:

```
UPDATE <View_name>
SET column name1 = value1, column name2 = value2…
WHERE <search_condition>
```

– דוגמא:

עדכון תאריך הלידה של עובד מספר 10 ב-View:

```
UPDATE emp_det_vw
set BirthDate = '1990-12-12'
where EmployeeID = 10
```

- תאריך הלידה יתעדכן גם ב-View וגם בטבלה המקורית Employees





# DELETE VIEW

- פקודת DELETE על View:

```
DELETE FROM <view_name>
WHERE <condition>
```

– דוגמא:

מחיקת עובד מספר 10 מה-View:

```
DELETE FROM emp_det_vw
WHERE EmployeeID = 10
```

- עובד מספר 10 ימחק גם ב-View וגם מהטבלה המקורית Employees

# תרגיל

- תרגיל 7 בחוברת תרגילים
- 30 דקות



15

# Stored Procedures

IITC - QA

המרכז הישראלי

לטכנולוגיות הבטחת איכות

www.iitc.co.il



16

# Stored Procedures

- אוסף מהודר (compiled collection) של הצהרות SQL המאוחסנות יחד כאובייקט בעל שם בודד בתוך מסד נתוני SQL Server
- פרוצדורות יכולות להכיל:
  - הצהרות SQL
  - הצהרות Transact SQL - הרחבה של Microsoft לשפת ה SQL, כגון: תמיכה במשתנים, בפונקציות, תנאים, לולאות וכד'




# Stored Procedures

- הפרוצדורות עוברות הידור פעם אחת.
- ניתן לקרוא לפרוצדורה מפרוצדורות אחרות
  - עד 32 רמות, פרוצדורה פנימית מכירה את האובייקטים שיצרה הפרוצדורה החיצונית
- ניתן להעביר פרמטרים לפרוצדורות

# Stored Procedures
# יתרונות

- הפרוצדורות מאפשרות שיתוף קוד עבור אפליקציות שונות
- כל שינוי מתבצע פעם אחת ואין צורך לעדכן כל אפליקציה בנפרד
- חוסך מהמשתמש להכיר את מבנה ה-database, הפעולות עבורו מבוצעות דרך הפרוצדורה
- משתמש יכול לקבל הרשאה להריץ פרוצדורה בעוד אין לו הרשאה לגשת לטבלאות ו-Views בבסיס הנתונים
- פרוצדורות משפרות ביצועים
- פרוצדורות מורידות עומס מהרשת שכן במקום לשלוח מספר הוראות כל אחת בנפרד ולקבל משוב על כל אחת מהן, נשלחת בקשה אחת ותגובה אחת




# יצירת פרוצדורה

- תחביר ליצירת פרוצדורה:

CREATE PROCEDURE שם הפרוצדורה AS קוד הפרוצדורה

- דוגמא:

*CREATE PROCEDURE getEmployees AS*
*Select * from employees*

- בסיום הכתיבה יש להריץ את הקוד על מנת לאחסן את הפרוצדורה בבסיס הנתונים
- הרצת הקוד יוצרת את הפרוצדורה ולא מריצה אותה!
  - נקבל הודעת "Command(s) completed successfully." המאשרת שהפרוצדורה נוצרה.

# הרצת פרוצדורה

- כדי להריץ את הפרוצדורה נקרא לה עם פקודת EXECUTE:
- תחביר להרצת פרוצדורה:

EXEC[UTE] שם הפרוצדורה

- דוגמא:

EXECUTE getEmployees

ניתן להריץ את הפרוצדורה גם ע"י כתיבת שמה בלבד (במידה ורק היא מורצת):

getEmployees





# מחיקת פרוצדורה

- תחביר למחיקת פרוצדורה:

DROP PROCEDURE שם הפרוצדורה

- דוגמא:

*DROP PROCEDURE getEmployees*

# שילוב פקודות DML בפרוצדורה

- שילוב פרוצדורה ופקודת INSERT:
  1. יוצרים את הפרוצדורה הרצויה
  2. משלבים את הפרוצדורה בפקודת Insert

דוגמא:

1. יצירת פרוצדורה השולפת את העובדים שהועסקו עד היום (לא כולל היום) מטבלת Employees:

```
CREATE PROCEDURE EmployeeCustomer
AS
SELECT upper(substring(LastName,1,4)+substring(FirstName,1,1)),
'Northwind Traders', rtrim(FirstName)+' '+LastName,'employee',Address, City ,Region ,
PostalCode , country ,('(206) 555-1234'+' x'+extension), NULL
FROM employees
WHERE HireDate<GETDATE()
```




# שילוב פרוצדורה ופקודת INSERT

2. הכנסת העובדים שנשלפים ע"י הפרוצדורה לטבלת Customers:

*INSERT INTO Customers*
*EXEC EmployeeCusromer* ← שם הפרוצדורה

- ניתן גם לבצע את פקודת ה-DML בפרוצדורה עצמה:

```
CREATE PROCEDURE InsertEmployee AS
INSERT INTO Employees (FirstName, LastName, Title, City)
Values ('Eddie', 'Vedder', 'Singer', 'Seattle')

EXEC InsertEmployee
```

# שינוי פרוצדורה

- תחביר:

ALTER PROCEDURE שם הפרוצדורה
AS
קוד הפרוצדורה החדש

- דוגמא:

Alter procedure InsertEmployee AS
INSERT INTO Employees (FirstName, LastName, Title, City)
Values ('Alex', 'Smith', 'Tester', 'London')




# פרמטרים בפרוצדורה

- ניתן לשלוח ערכים לפרוצדורה על ידי שימוש בפרמטרי INPUT
- ניתן לקבל חזרה ערכים על ידי שימוש בפרמטרי OUTPUT

# פרמטרים בפרוצדורה
# INPUT

- הגדרת פרוצדורה המקבלת פרמטרים INPUT:
  - פרוצדורה תקבל פרמטרים לאחר הגדרת שמה (הפרמטרים יכולים להופיע בסוגריים אך אין זו חובה)
  - לפני שם כל פרמטר יופיע הסימן @ ולאחריו יצוין סוג ה DataType שלו
  - הפרמטרים יכולים לקבל ערך ברירת מחדל ואף ערך NULL במידה והוגדר
  - ערך ברירת מחדל לפרמטר - המשתמש לא יהיה חייב לשלוח ערך עבור פרמטר זה
  - פרמטר מסוג זה הוא פרמטר ברירת המחדל
- תחביר הגדרת פרמטר INPUT:

@parameter_name data_type [=default]




# פרמטרים בפרוצדורה
# INPUT

- **דוגמא 1**: פרוצדורה המקבלת שני פרמטרים של תאריך ומחזירה את ההזמנות שתאריך המשלוח שלהם בין טווח הערכים שהתקבל בפרמטרים:

```
CREATE PROCEDURE ShippedByDate
( @beginningDate dateTime,
@EndingDate DateTime)
As
IF @BeginningDate Is NULL OR @EndingDate IS NULL
Begin
PRINT('Null Values are not Allowed')
RETURN
End
SELECT ShippedDate, OrderId
FROM Orders
WHERE shippedDate between @beginningDate and @endingDate
```

הגדרת הפרמטרים של הפרוצדורה

ריצת הפרוצדורה תסתיים

# פרמטרים בפרוצדורה
# INPUT

– הרצת הפרוצדורה עם ערכים תקינים תחזיר את הרשומות שעונות על התנאי בפרוצדורה:

*EXEC ShippedByDate '12-01-96','12-31-96'*

– הרצת הפרוצדורה עם ערך לא תקין תחזיר הודעה מתאימה (לפי הקוד של הפרוצדורה):

EXEC ShippedByDate '12-01-96',NULL

– התוצאה:

Null Values are not Allowed





# פרמטרים בפרוצדורה
# INPUT

- **דוגמא 2**: פרוצדורה המוסיפה לקוח חדש לטבלת הלקוחות:

```
CREATE PROCEDURE AddCustomer
(
@CustomerId nchar(5),  @CompanyName nvarchar(40),
@ContactName nvarchar (30) =NULL, @ContactTitle nvarchar (30) =NULL,
@Address nvarchar (60) =NULL, @City nvarchar (15) =NULL,
@Region nvarchar (15)= NULL, @PostalCode nvarchar (10)= NULL, @Country nvarchar
(15)= NULL, @Phone nvarchar (24) =NULL, @Fax nvarchar (24)= NULL
)
AS
INSERT INTO Customers
VALUES (@CustomerId, @CompanyName,@ContactName,@ContactTitle, @Address,
@City, @Region, @PostalCode, @Country, @Phone,@Fax)
```

# פרמטרים בפרוצדורה INPUT

- הרצת הפרוצדורה:
  - דרך א': ציון הערכים לפי סדר הפרמטרים:

EXEC AddCustomer 'GRAM','Alfreds Futterkiste' , 'Maria Anders', 'Sales Representive','Obere Str .57','Berlin',Null,12209,'Germany','030-0074321',Null

- דרך ב': ציון הפרמטרים אליהם מועברים הערכים ולא בהכרח כל הערכים (במידה ומוגדר להם ערך ברירת מחדל):

*EXEC AddCustomer*
*@CustomerId='ALFAKI', @ContactName='Maria Anders',*
*@CompanyName='Alfreds Futterkiste', @ContactTitle = 'Sales Representive',*
*@Address= 'Obere Str .57', @City = 'Berlin', @Region = Null, @PostalCode = 12209,*
*@Country = 'Germany', @Phone = '030-0074321',* @Fax = Default

שימוש בערך ברירת מחדל שהוגדר




# החזרת פרמטרים מפרוצדורה OUTPUT

- פרוצדורה יכולה להחזיר פרמטרים לתכנית הקוראת
- פרמטר מוחזר (OUTPUT) יוגדר כמו פרמטר ה-INPUT רק שלאחר ציון הסוג שלו (data Type) תצוין המילה OUTPUT
- תחביר:

@parameter_name data_type OUTPUT

# החזרת פרמטרים מפרוצדורה OUTPUT

- דוגמא: פרוצדורה המקבלת שני מספרים ומחשבת את מכפלתם:

```
CREATE PROCEDURE MathMult
@m1 smallint,
@m2 smallint,
@result smallint OUTPUT
AS
SET @result=@m1*@m2
```





# הרצת פרוצדורה המחזירה ערך

- מאחר והפרוצדורה מחזירה ערך, **חובה להגדיר משתנה אשר יקבל את הערך אותו מחזירה הפרוצדורה**
- ניתן להדפיס את הערך החוזר על ידי שימוש ב- SELECT או PRINT:

*הצהרה על משתנה אשר יקבל את הערך המוחזר:*

```
DECLARE @answer smallint
```

*הרצת הפרוצדורה:*

```
EXEC MathMult 5,6,@answer OUTPUT
```

*הדפסת הפרוצדורה:*

```
SELECT 'The result is : ' ,@answer
```

יש להריץ את פקודת ההצהרה על המשתנה ופקודות הרצת הפרוצדורה והדפסת התוצאה - ביחד

# RETURN

- ניתן להחזיר ערכים כקוד החזרה מפרוצדורה ע"י שימוש במילה RETURN
- יש להימנע מהחזרת קוד NULL ומספרים שליליים שכן אלו הם קודי שגיאה שמורים של המערכת
- **דוגמא**: פרוצדורה הבודקת האם לקוח מסוים קיים בטבלת לקוחות
  - אם קיים: מחזירה 1
  - אם לא קיים: מחזירה 0

```
CREATE PROCEDURE checkCust
@CuID nchar(5),
@CuCompName nvarchar(40)
AS
IF EXISTS (SELECT *
        FROM customers
        WHERE CustomerID=@CuID AND CompanyName=@CuCompName)
RETURN (1)
ELSE
RETURN(0)
```





# RETURN

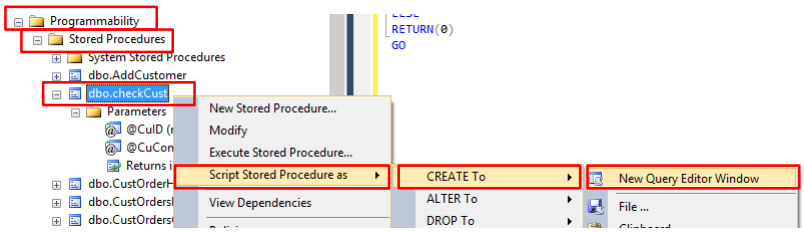
- הרצת הפרוצדורה מחייבת הגדרת משתנה שיקלוט את הערך המוחזר, הפניה אליו בעזרת משפט SELECT:

```
DECLARE @returnCode int
EXEC @returnCode=CheckCust 'DAVON', 'Northwind Traders'
SELECT @returnCode
```

  - התוצאה שהתקבלה: 1

# צפיה ב-Stored Procedures המוגדרים ב-DB

- ניתן לראות את הפרוצדורות המוגדרות ב-DB ב- SQL Server Management Studio ע"י:
  - פתיחת תיקיית Programmability -> Stored Procedures
  - מציאת את הפרוצדורה הרצויה וסימונה
  - כפתור ימני -> Script Stored Procedure as -> CREATE To -> New Query Editor Window








# תרגיל

- תרגיל 8 בחוברת תרגילים
- 30 דקות

# SQL
# בדיקות DB

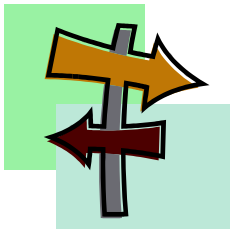
IITC - QA
המרכז הישראלי
לטכנולוגיות הבטחת איכות
www.iitc.co.il





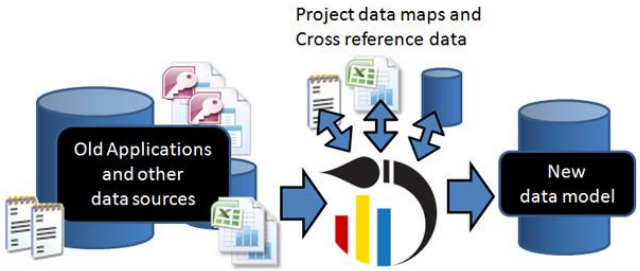
## נושאים

- בדיקות הסבות נתונים
- DB Auditing
- DB Log Files
- בדיקות DB ואיתור באגים



# בדיקות הסבות נתונים
# Data Migration

- בדיקות הסבות נתונים מתבצעות כאשר מבנה ה-DB במוצר/גירסה החדשה שונה ממבנה ה- DB של המוצר/הגירסה הקודמת.





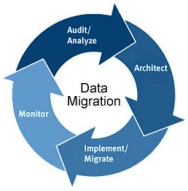


# בדיקות הסבות נתונים

- שינויי DB יכולים לכלול:
  - הוספת טבלאות חדשות
  - מחיקת טבלאות קיימות
  - שינויי מבנה בטבלאות קיימות (הוספת עמודה/הורדת עמודה/שינויים במפתחות ראשיים)
  - שינויים בקישורים בין הטבלאות
  - שינויים באילוצים של טבלאות ושדות
  - המרות נתונים מסוג נתון אחד לסוג נתון אחר

# בדיקות הסבות נתונים

- צוות הבדיקות צריך לקבל את רשימת השינויים במבנה ה-DB מהצוות/ים הרלוונטיים
- הבדיקות כוללות:
  - בדיקת תהליך ההמרה
    - השוואת מבנה הטבלה לפני ואחרי השינוי
    - בדיקת דוחות תסריטי המרת הנתונים הכוללים את פקודות DML (insert, update, delete) של תהליך ההמרה
      - למשל Conversion.Script.Extract.txt








# בדיקות הסבות נתונים

- בדיקת איכות הנתונים:
  - השוואת רשומות נתונים מהמערכת הישנה עם הרשומות התואמות להן במערכת החדשה
  - בדיקת ערכים וטיפוסי נתונים ישנים הממופים לערכים וטיפוסי נתונים חדשים (למשל VARCHAR לעומת NVARCHAR או BLOB)
  - בדיקת כמות הרשומות בטבלה לפני ואחרי המרת הנתונים
  - בדיקת רשומות בן חסרות:
    - עבור כל רשומת אב שלה צריכות להיווצר רשומות בן חדשות בתהליך הסבת הנתונים – יש לוודא שהן נוצרו כמצופה בנתונים המוסבים

# בדיקות הסבות נתונים

– בדיקת פונקציונאליות המערכת החדשה עם נתונים מוסבים
- בדיקת כל חלקי המערכת עם נתונים מוסבים על מנת לוודא עבודה תקינה עם הנתונים
- בד"כ כחלק מהבדיקות הפונקציונאליות, דרך ה-GUI

- תמיכת כלים בבדיקות הסבת נתונים:
  – TRUcompare – משווה בין נתוני המקור לנתונים המוסבים (לא חינמי)





# DB Auditing

- מעקב אחר השימוש ב-DB ע"י משתמשים, על מנת לוודא שמשתמשים ללא הרשאות מתאימות לא יוכלו להגיע לנתונים רגישים
- ישנם סטנדרטים שונים של אבטחת מידע במאגרי נתונים:
  – HIPAA – להגנה על המידע במערכות רפואיות
  – FCRA – להגנה על המידע של צרכנים (אשראי)
  – PCI – להגנה על המידע של צרכנים (פרטי כרטיסי אשראי)
  – ISO 17799 (Basel II) - להגנה על לקוחות בנקים
  – Sarbanes-Oxley Act – להגנה על מידע פיננסי
  – ועוד

# DB Auditing

- מטרת ה- Audit - לבדוק עמידה בסטנדרטים רלוונטיים בעיקר לגבי:
  - פעילויות ניהול - Administrative activity
  - פעילויות כניסה ויציאה - Logon and logoff activity
  - כשלים - Failures
  - שימוש בהרשאות מערכת - Use of system privileges
  - שימוש בהרשאות "all" - Use of “all” privileges
  - שינויים במבנה ה-DB
  - גישה לאוביקטים קריטיים



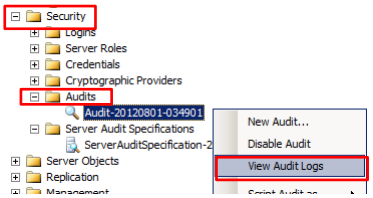

# DB Auditing

- ישנן 2 רמות של Auditing:
  - רמת SQL Server Audit - מספק כלים ותהליכים המאפשרים להפעיל, לשמור ולצפות ב-Audits על שרתים ואובייקטים שונים
  - רמת Database audit - למעקב אחר ארועים ב-DB
- ארועים עליהם עושים מעקב יכולים להרשם ב-Event Logs או בקבצי Audit

# Viewing Audit Log Files

- דרך SQL Management Studio:

– בתיקיית Security סמן את שם ה-Audit תחת תיקיית Audits - > כפתור ימני בעכבר -> View Audit Logs




– דרך פקודת T-SQL:

SELECT * FROM sys.fn_get_audit_file ('<File path & name.sqlaudit>',default,default)

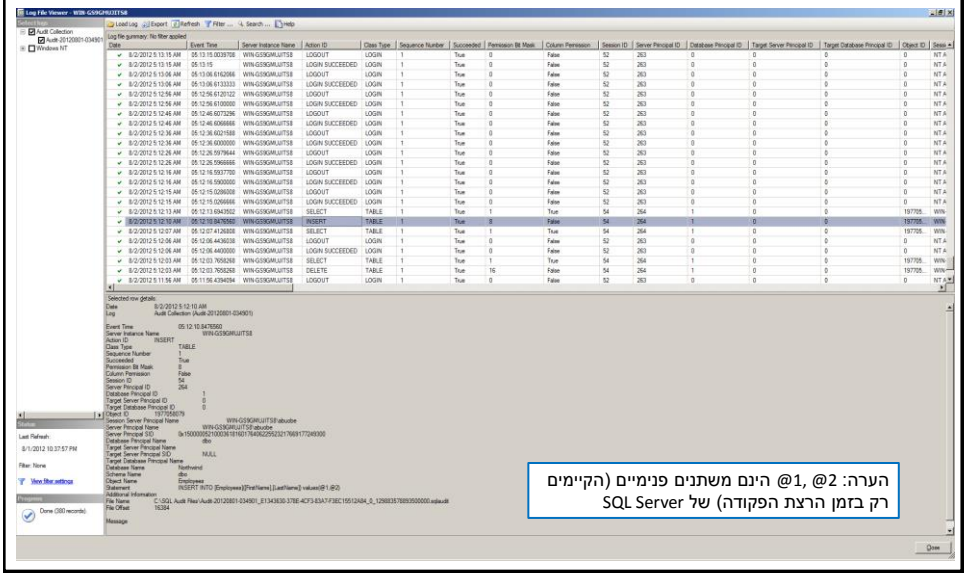
- על מנת לראות את הפעולות שאינן Login, Logout יש להוסיף:

where object_id <>0



18

# Viewing Audit Log Files



הערה: @2, @1 הינם משתנים פנימיים (הקיימים רק בזמן הרצת הפקודה) של SQL Server

# Server & DB Audit
# לקריאה נוספת

- פירוט והסבר על קבוצות הפעולות:

http://msdn.microsoft.com/en-us/library/cc280663.aspx

- הסברים נוספים באתר msdn:

http://msdn.microsoft.com/en-us/library/cc280525.aspx
http://msdn.microsoft.com/en-us/library/cc280424.aspx

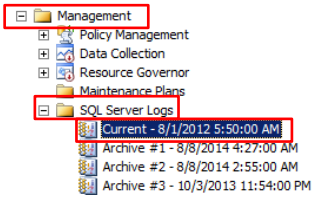



# SQL Server Log Files

- ניתן להשתמש בקבצי LOG של SQL Server על מנת לראות פרטים לגבי:
  - נתונים הנרשמים בקבצי ה-Audit
  - Errors
  - פעולות הקשורות ל-SQL Server
  - פעולות הקשורות ל-SQL Server Agent
- על מנת לראות קבצי Log של SQL Server עלינו להיות בעלי הרשאה מתאימה
  - חברים בקבוצת securityadmin בתיקיית Server Roles תחת תיקיית Security

# View SQL Server Log Files

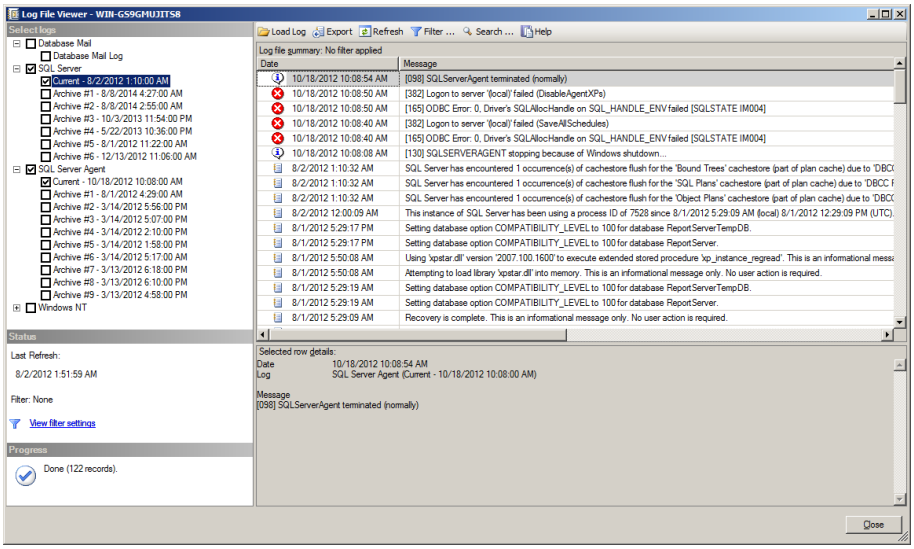
- על מנת לצפות ב- SQL Server Log Files דרך SQL Management Studio:
  - בתיקיית Management פתח את תיקיית SQL Server Logs -> סמן את שם ה-Log הרצוי ולחץ עליו פעמיים





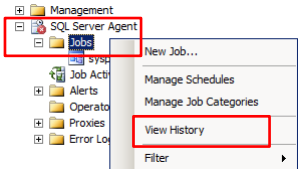



# Log File Viewer

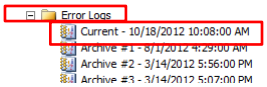


# View SQL Server Agent Log Files

- על מנת לצפות ב- SQL Server Agent Log Files דרך SQL Management Studio:
  - **דוח Jobs:** בתיקיית SQL Server Agent סמן את תיקיית Jobs-> כפתור ימני -> View History




  - **דוח Errors:** בתיקיית SQL Server Agent פתח את תיקיית Error Logs -> סמן את שם ה-Log הרצוי ולחץ עליו פעמיים








# צפיה בדוח שגיאות ע"י Stored Procedure

- אפשר להשתמש ב-SP פנימי של SQL Server המראה את ה-Error Log File :

EXEC **sp_readerrorlog** <Err Log #>, <log type>, <Search string>

  - הפרמטרים האפשריים:

**Value of error log file you want to read:** 0 = current, 1 = Archive #1, 2 = Archive #2, etc...
**Log file type:** 1 or NULL = error log, 2 = SQL Agent log
**Search string 1:** String one you want to search for
**Search string 2:** String two you want to search for to further refine the results

  - דוגמא: EXEC sp_readerrorlog 6, 1, '2005'

יראה את Archived Error Log File # 6, רק את השורות עם המחרוזת "2005"

# בדיקות Database ואיתור באגים

- בדיקת שלמות הנתונים ונכונות הנתונים
- Normalization - בדיקת מבנה הטבלאות
- בדיקת חסיון מידע: סיסמאות ומידע חסוי צריכים להיות מוצפנים
- בדיקת זמני עיבוד שאילתות שונות
- Multi-user support – תמיכה במספר משתמשים בו-זמנית המבצעים פעולות על אותן טבלאות





# בדיקת שלמות הנתונים

- מבטיחות שהנתונים המאוחסנים ב-DB הינם שלמים ונכונים
- מבטיחות כי שינוי ותחזוקה של הנתונים מתבצעים לפי מודל הנתונים (Data Model) ולפי סוגי הנתונים המוגדרים בטבלאות
  - דוגמא: על מנת לשמור על תקינות הנתונים, שדה מספרי לא יקבל אותיות

# Normalization

- תהליך אירגון השדות והטבלאות בצורה יעילה ב-DB
  - מניעת כפילויות בנתונים
  - אחסון נתונים קשורים בטבלה
  - הגדרת הקשרים בין הטבלאות (FK ,PK)
  - המטרה: הוספת/מחיקת/עדכון נתון יעשה רק בטבלה אחת וע"י הקשרים בין הטבלאות יתפשט לשאר הטבלאות
- עקרונות ה-Normalization מחולקים ל-4 Normal Forms:
  - First Normal Form
  - Second Normal Form
  - Third Normal Form
  - Boyce Codd Normal Form

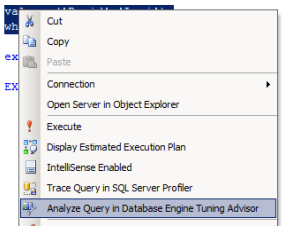


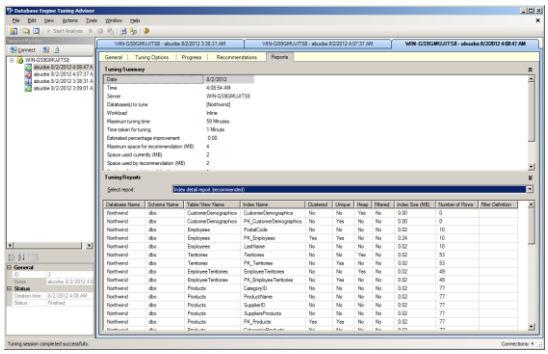

# מבנה טבלאות
# מה עוד בודקים?

- שמות טבלאות, שמות שדות – עמידה במוסכמות Naming conventions
- סוג הנתונים בכל שדה הינו הנכון לפי התכנון של ה-DB
  - בדיקה גם של סוגי הנתונים בשדות FK לפי שדה ה-PK שלהם
- אורך השדות הינו לפי תכנון ה-DB
  - בדיקה גם של אורך השדות בשדות FK לפי שדה ה-PK שלהם
- קיום PK, FK בטבלאות
- בדיקת אילוצי Check, Default בטבלאות
- קיום אינדקס על עמודות בטבלאות

# כלי עזר לבדיקות DB

- **MS Database Engine Tuning Advisor**
  - בודק את יעילות השאילתה, ומציג המלצות שיובילו לשיפור זמני עיבוד השאילתה
  - ניתן להפעילו דרך SQL Server Management Studio:
    - לאחר כתיבת השאילתה -> כפתור ימני -> Analyze Query in Database Engine Tuning Advisor



# כלי עזר לבדיקות DB

- DB Optimizer של חברת Embarcadero:

https://www.embarcadero.com/products/db-optimizer

  - בודק זמני תגובה של ה-DB
  - שיפור פקודות SQL לשיפור ביצועים
- Datatect של חברת Banner Software:

http://www.softpedia.com/get/Internet/Servers/Server-Tools/Datatect.shtml

  - מייצר נתונים בבסיסי נתונים: Oracle, Sybase, SQL Server, and Informix

# כלי עזר לבדיקות DB

- DTM DB Stress

http://www.sqledit.com/stress/

  - לבדיקות עומסים על DB

- DBMonster

http://dbmonster.sourceforge.net/

  - ליצירת נתונים והעמסת DB
  - לשיפור שאילתות ויעילות זמני תגובה
  - לשיפור מבנה ה-DB





# לקריאה נוספת

- http://databases.about.com/od/specificproducts/a/normalization.htm
- http://en.wikipedia.org/wiki/Database_normalization

# תרגילים

# Views, Stored Procedures

# 1. Views – תרגיל

1. מה יקרה ל-View כאשר נמחק טבלה בה הוא תלוי?
2. האם ניתן למחוק View שנוצר ממספר טבלאות?
3. האם הפקודות הבאות נכונות?

a.
```
CREATE VIEW Credit_Debt
AS
SELECT *
FROM Debt
WHERE account_id = 4
```
b.
```
CREATE UNIQUE VIEW Debts
AS
SELECT *
FROM Debt_tbl
```
c.
```
DROP * FROM VIEW Debts
```

4. האם ניתן ליצור VIEW המתבסס על VIEW אחר?

**ענה על השאלות הבאות בעזרת בבסיס הנתונים Northwind (בתוכנת SQL Server):**

5. צור VIEW בשם CurrentProducts_view המציג את רשימת המוצרים האקטיביים (Discontinued = 0) מתוך טבלת Products. ה- View יציג את העמודות ProductID, ProductName בלבד. הרץ את ה-View וודא שהוא עובד כנדרש.
6. צור VIEW בשם HighPrice_View המציג את המוצרים שמחירם מעל המחיר הממוצע, מתוך טבלת Products. ה- View יציג את העמודות UnitPrice, ProductName בלבד.
7. צור VIEW בשם TotalSales97_View המציג את סכום המכירות עבור כל קטגוריה של מוצרים שנמכרו בשנת 1997. ה- VIEW יציג את העמודות CategoryName, סכום המכירות בקטגוריה (קרא לעמודה זו בשם Alias: CategorySales).
   a. ה- VIEW יקח את הנתונים מתוך VIEW קיים, בשם [Product Sales for 1997]
   b. כל קטגוריה תופיע פעם אחת בלבד (השתמש ב-Group By)
   c. הרץ שאילתה המציגה רק את סכום המכירות של קטגוריה 'Beverages' מתוך ה- VIEW : TotalSales97
8. עדכן את ה- VIEW: CurrentProducts_view שיצרת בסעיף 5 בתרגיל זה, כך שיכיל גם את עמודת CategoryID.
   a. הרץ שאילתה המציגה את הנתונים עבור קטגוריות מספר 2,4,8 בלבד מתוך ה-VIEW: CurrentProducts_view
   b. הרץ שאילתה המציגה את כמות המוצרים בכל קטגוריה, עבור קטגוריות מספר 2,4,8 בלבד מתוך ה-VIEW: CurrentProducts_view
9. מחק את ה-VIEW: CurrentProducts_view וודא שהוא נמחק.



www.iitc.co.il אתר: info@iitc.co.il: Email 077-4123211/2 טלפון:

**התרגילים הבאים מתייחסים ל Prods :VIEW ותלויים אחד בשני:**

10. צור VIEW בשם Prods המציג את העמודות הבאות: ProductID, ProductName, UnitPrice, ההפרש בין סך כל הפריטים במלאי ( UnitsInStock) לסך כל הפריטים המוזמנים (UnitsOnOrder), קרא לעמודה זו: UnitsLeft. יש לכלול רק מוצרים שעולים פחות מ 30 .
11. פתח טרנזאקציה ובצע את הפעולות הבאות על VIEW: Prods:
    a. עדכן את שם מוצר מספר 52 ל: 'Jaffa'
    b. עדכן את מחיר מוצר מספר 52 ל 31
    c. הצג את פרטי מוצר 52 מה VIEW: Prods. האם הצלחת? מדוע?
    d. אל תשמור את השינויים (RollBack)
12. עדכן את VIEW: Prods כך שיכלול בדיקה על התנאי (With Check Option)
13. בצע שוב את ההנחיות שבסעיף 11. האם הצלחת לעדכן את מחירו של מוצר 52? מדוע?
14. מחק את ה-View: Prods



www.iitc.co.il אתר: info@iitc.co.il: Email **077-4123211/2**טלפון:

# 2. Stored Procedures – תרגיל

1. מה הן היתרונות בשימוש ב-Stored Procedures?
2. מה ההבדל בין VIEWS ל-Stored Procedures?

**ענה על השאלות הבאות בעזרת בבסיס הנתונים Northwind (בתוכנת SQL Server):**

3. צור פרוצדורה בשם Prod_det המציגה עבור מוצר מספר 6 את שמו ושם הקטגוריה אליה הוא שייך. הנתונים ילקחו משתי טבלאות (השתמש ב-JOIN):
   a. Products
   b. Categories

   הרץ את הפרוצדורה וודא פעולה תקינה.
4. שנה את הפרוצדורה מסעיף 3, כך שהיא תקבל מספר מוצר שונה בכל הרצה.
5. צור פרוצדורה בשם Cust_det המקבלת קוד לקוח (שים לב: עמודת CustomerID היא מחרוזתית) ומציגה את הפרטים הבאים מתוך טבלת Customers: Address, Country, CompanyName
6. צור פרוצדורה בשם Cust_Orders המקבלת קוד לקוח כפרמטר נכנס ומחזירה את סך כל ההזמנות שהלקוח ביצע כפרמטר יוצא.
   a. הפרוצדורה תדפיס לחלון התוצאות את סך כל ההזמנות של הלקוח.
   b. קרא לפרוצדורה שיצרת עם קוד לקוח: 'ALFKI' ו-'BERGS'.
7. פתח את ה-Stored Procedure המובנה בתוך Northwind (כחלק מה-Northwind sample DB) שנקרא:

   dbo.Sales By Year בתוך חלון שאילתה חדש.
   a. מה עושה פרוצדורה זו?
   b. הרץ את הפרוצדורה עם ערכים ואלידים (מתוך הטבלה) ועם ערכים לא ואלידים (שאינם קיימים בטבלה) ובדוק שהיא עובדת כנדרש.

# פתרונות תרגילים

# Views, Stored Procedures

# 3. פתרונות לתרגיל – Views

15. מה יקרה ל-View כאשר נמחק טבלה בה הוא תלוי?

**תשובה**: ה-View לא יעבוד, מכיון שהנתונים עליהם הוא נשען אינם קיימים עוד.

16. האם ניתן למחוק View שנוצר ממספר טבלאות?

**תשובה**: לא ניתן למחוק View שנוצר ממספר טבלאות. ניתן למחוק רק View שנוצר מטבלה אחת בלבד.

17. האם הפקודות הבאות נכונות?

a.
```
CREATE VIEW Credit_Debt
AS
SELECT *
FROM Debt
WHERE account_id = 4
```

**תשובה**: כן, המשפט נכון.

b.
```
CREATE UNIQUE VIEW Debts
AS
SELECT *
FROM Debt_tbl
```

**תשובה:** לא. מילת המפתח UNIQUE אינה חלק מהתחביר של פקודת יצירת VIEW.

c.
```
DROP * FROM VIEW Debts
```

**תשובה:** לא. התחביר הנכון למחיקת VIEW הינו:

```
DROP VIEW Debts
```

18. האם ניתן ליצור VIEW המתבסס על VIEW אחר?

**תשובה**: כן. בד"כ יוצרים VIEWהמתבסס על טבלה, אך ניתן גם ליצור VIEW המתבסס על VIEW אחר.



www.iitc.co.il אתר: info@iitc.co.il: Email **077-4123211/2**טלפון:

**ענה על השאלות הבאות בעזרת בבסיס הנתונים Northwind (בתוכנת SQL Server):**

```
--5

CREATE VIEW CurrentProducts_view AS
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued= 0


select * from CurrentProducts_view

--6
CREATE VIEW HighPrice_View2 AS
SELECT ProductName,UnitPrice
FROM Products
WHERE UnitPrice>(SELECT AVG(UnitPrice) FROM Products)

Select * from HighPrice_View

--OR Find out the average unitprice and then create the view (if you don't know
subqueries yet):
select avg(unitprice) from products

CREATE VIEW HighPrice_View2 AS
SELECT ProductName,UnitPrice
FROM Products
WHERE UnitPrice>32.445

Select * from HighPrice_View2

--7

CREATE VIEW TotalSales97_View AS
SELECT CategoryName,Sum(ProductSales) AS CategorySales
FROM [Product Sales for 1997]
GROUP BY CategoryName

select * from TotalSales97_View

--7c
select * from TotalSales97_View
where CategoryName = 'Beverages'

--8
Alter view currentProducts_view AS
SELECT ProductID,ProductName, CategoryID
FROM Products
WHERE Discontinued= 0

select * from CurrentProducts_view
--8a
select * from CurrentProducts_view
where CategoryID in (2,4,8)
```

```
--8b
select CategoryID, count(productid) AS NumOfProducts
from CurrentProducts_view
where CategoryID in (2,4,8)
group by CategoryID

--9
drop view CurrentProducts_view

select * from CurrentProducts_view

--10
CREATE VIEW Prods
AS
SELECT ProductID, ProductName, UnitPrice, UnitsInStock-UnitsOnOrder as UnitsLeft
FROM Products
WHERE UnitPrice < 30

select * from Prods

--11
select * from Prods
where ProductID=52


begin transaction
--11a
UPDATE prods
SET ProductName = 'Jaffa'
WHERE ProductID=52

--11b
update prods
SET UnitPrice = 31
where ProductID = 52

--11c
select * from Prods
where ProductID=52

--אנו לא מצליחים לראות את פרטי מוצר 52 המעודכנים מכיון מחירו החדש גבוה מ 30 --

--11d
rollback

--12
ALTER VIEW Prods
AS
SELECT ProductID, ProductName, UnitPrice, UnitsInStock-UnitsOnOrder as UnitsLeft
FROM Products
WHERE UnitPrice < 30
with check option
```

```
--13
select * from Prods
where ProductID=52

begin transaction
--13a
UPDATE prods
SET ProductName = 'Jaffa'
WHERE ProductID=52

--13b
update prods
SET UnitPrice = 31
where ProductID = 52

--13c
select * from Prods
where ProductID=52

--אנו לא מצליחים לעדכן את מחיר מוצר 52 מכיון שהמחיר החדש אינו עומד בתנאי הבדיקה

--13d
rollback

--14
drop view prods
```

# 4. Stored Procedures – פתרון תרגיל

8. מה הן היתרונות בשימוש ב-Stored Procedures?

**תשובה:**

- הפרוצדורות מאפשרות שיתוף קוד עבור אפליקציות שונות
- כל שינוי מתבצע פעם אחת ואין צורך לעדכן כל אפליקציה בנפרד
- חוסך מהמשתמש להכיר את מבנה ה-database, הפעולות עבורו מבוצעות דרך הפרוצדורה
- משתמש יכול לקבל הרשאה להריץ פרוצדורה בעוד אין לו הרשאה לגשת לטבלאות ו-Views בבסיס הנתונים
- פרוצדורות משפרות ביצועים
- פרוצדורות מורידות עומס מהרשת שכן במקום לשלוח מספר הוראות כל אחת בנפרד ולקבל משוב על כל אחת מהן, נשלחת בקשה אחת ותגובה אחת

9. מה ההבדל בין VIEWS ל-Stored Procedures?

**תשובה:**

**VIEW** הינו טבלה וירטואלית המתבסת על טבלה/טבלאות/Views אחרים. הנתונים המוצגים ב-View אינם תופסים מקום בזיכרון כחלק מה-VIEW, אלא רק הגדרת ה-VIEW תופסת מקום. הנתונים אינם מאוחסנים בצורה קבועה ב-VIEW עצמו. View אינו מקבל פרמטרים. ה- VIEW משמש לראות נתונים מטבלאות. משמש לצרכי אבטחת מידע וגישה למידע רגיש. ניתן להגדיר VIEW השולף רק את המידע שאנו מעוניינים לחשוף למשתמש מסוים, ולא לתת למשתמש זה הרשאה לראות את הטבלאות עצמן.

**Stored Procedure** הינה קבוצה של פקודות SQL שניתן לשמור ב- DB ולשתף בין מספר משתמשים ברשת. ה-SP מקומפל פעם אחת בלבד ע"י שרת ה-DB, וניתן להריצו מספר רב של פעמים. ה- SP מקבל פרמטרים ויכול גם להחזיר פרמטרים. SP משמשים לשינוי והכנסת נתונים לטבלאות.

**ענה על השאלות הבאות בעזרת בבסיס הנתונים Northwind (בתוכנת SQL Server):**

```
--3
CREATE PROCEDURE prod_det
AS
SELECT p.productname, c.categoryname
FROM products p JOIN categories c
ON p.categoryid = c.categoryid
WHERE p.productid = 6

-- הפרוצדורה של הרצה
EXEC prod_det
```

```
--4
ALTER PROCEDURE prod_det
(@prod_num INT)
AS
SELECT p.ProductName, c.CategoryName
FROM products p JOIN categories c
ON p.CategoryID = c. CategoryID
WHERE p.ProductID = @prod_num

-- הפרוצדורה של הרצה
EXEC prod_det 19

--5
CREATE PROCEDURE cust_det
(@cust_id NCHAR(5))
AS
SELECT CompanyName, Country, Address
FROM customers
WHERE CustomerID = @cust_id

--הפרוצדורה של הרצה--
EXEC cust_det 'ALFKI'

--6
CREATE PROCEDURE cust_ord_num
(@cust_id NCHAR(5), @ord_num INT OUTPUT)
AS
SELECT @ord_num = COUNT(OrderID)
FROM orders
WHERE CustomerID = @cust_id

--6b - הפרוצדורה של הרצה
DECLARE @orders_amount INT
EXEC cust_ord_num 'ALFKI' ,@orders_amount OUTPUT
PRINT @orders_amount

DECLARE @orders_amount INT
EXEC cust_ord_num 'BERGS' ,@orders_amount OUTPUT
PRINT @orders_amount

--7
--7a
      --הפרוצדורה מקבלת 2 תאריכים המייצגים טווח של תאריכי משלוח הזמנה ושולפת פרטי ההזמנות שנשלחו בטווח
תאריכים זה
--הנתונים נשלפים מטבלת Orders ומ VIEW- "Order Subtotals"

--7b
--הרצה עם נתונים קיימים בטבלה--
exec [Sales by Year] '1996-07-17','1996-08-20'

--הרצה עם נתונים שאינם קיימים בטבלה--
exec [Sales by Year] '2000-07-17','2000-08-20'
```