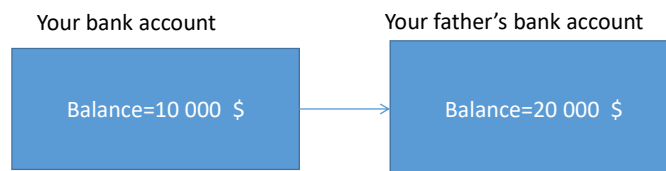


Database I.

Etelka Szendrői Dr. (PhD)

Transactions

- What is a transaction?



Transfer 5 000 \$ from your account to your Father's account:

- Debit 5 000 \$ from your bank account (Subtract 5 000 \$)
- Credit 5 000 \$ to your father's bank account (Add 5 000 \$)

Transactions

What is a transaction?

- A transaction is a sequence of one or more SQL operations treated as a unit
- Also known as a Unit of Work (UOW)
- Transactions appear to run in isolation
- If the system fails, each transaction's changes are reflected either entirely or not at all
- Transactions ensure that multiple data modifications are processed together or not at all
- The transaction log ensures that updates are complete and recoverable
- Transactions use locks

3

Transactions

- A transaction starts with any **SQL statement** and ends with a **COMMIT** or **ROLLBACK**
- COMMIT statement makes changes permanent to the database
- ROLLBACK statement reverses changes
- COMMIT and ROLLBACK statements release all locks

4

Example of transactions

```
INSERT INTO emp VALUES (100, 'John')
INSERT INTO emp VALUES (200, 'Mary')
COMMIT
```

First SQL statement
starts transaction

EmpID	Name
100	John
200	Mary

No changes applied due to
ROLLBACK

```
DELETE FROM emp WHERE name='Mary'
UPDATE emp SET EmpID=101
WHERE name='John'
ROLLBACK
```

EmpID	Name
100	John
200	Mary

```
UPDATE emp SET name='Peter'
WHERE EmpID=100
COMMIT
```

EmpID	Name
100	Peter
200	Mary

```
ROLLBACK
```

There is nothing to
rollback



5

Transactions – ACID rules

• Atomicity

- All statements in the transaction are treated as a unit.
- If the transaction completes successfully, everything is committed
- If the transaction fails, everything done up to the point of failure is rolled back.

• Consistency

- Any transaction will take the data from one consistent state to another, so only valid consistent data is stored in the database

• Isolation

- Concurrent transactions cannot interfere with each other

• Durability

- Committed transactions have their changes persisted in the database

6

Concurrency & Locking

7

What Are Locks?

- **Two main types of lock:**
 - Read locks – allow others to read but not write
 - Write locks – stop others reading or writing
- **Deadlocks can occur**
- **Locks prevent update conflicts**
 - Locking ensures that transactions are serialized
 - Locking is automatic
 - Locks enable concurrent use of data

What Is Concurrency Control?

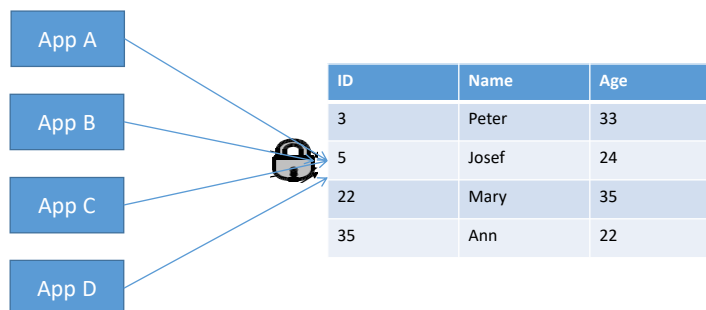
• Pessimistic

- Locks data when data is read in preparation for an update
- Other users are blocked until lock is released
- Use where high contention for data exists

• Optimistic

- Locks data when an update is performed
- Error received if data was changed since initial read
- Use when low contention for data exists

Concurrency and Locking



- **Concurrency:**
 - Multiple users accessing the same resources at the same time
- **Locking:**
 - Mechanism to ensure data integrity and consistency

Locking

- Locks are acquired automatically as needed to support a transaction based on “isolation levels”
- COMMIT and ROLLBACK statements release all locks
- Two basic types of locks:
 - Share locks (S locks) – acquired when an application wants to read and prevent others from updating the same row
 - Exclusive locks (X locks) – acquired when an application updates, inserts, or deletes a row

11

Managing transactions

Autocommit Transactions

- Default transaction mode
- Every statement is committed or rolled back when it has completed
 - If it completes successfully – it is committed
 - If it fails – it is rolled back
- Compile errors result in a batch not being executed

Explicit Transactions

- BEGIN TRANSACTION
- COMMIT TRANSACTION
- ROLLBACK TRANSACTION

```
BEGIN TRANSACTION fund_transfer
EXEC debit_checking '100', 'account1'
EXEC credit_savings '100', 'account2'
COMMIT TRANSACTION
```

- SAVE TRANSACTION
- Transaction log

Implicit Transactions

- Setting implicit transaction mode on

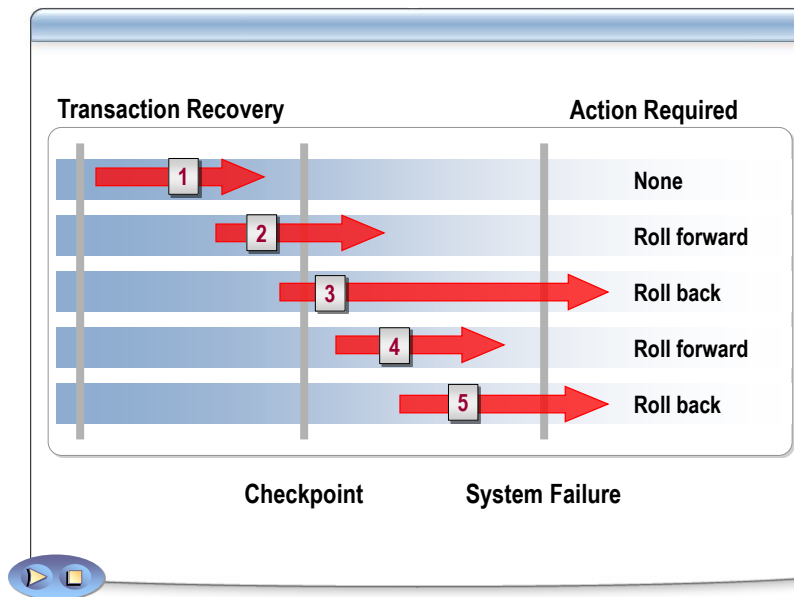
```
SET IMPLICIT_TRANSACTION ON
```

- An implicit transaction starts when one of the following statements is executed

- ALTER DATABASE
- CREATE
- DELETE
- DROP
- FETCH
- GRANT
- INSERT
- OPEN
- REVOKE
- SELECT
- TRUNCATE TABLE
- UPDATE

- Transaction must be explicitly completed with **COMMIT** or **ROLLBACK TRANSACTION**

Transaction Recovery



Considerations for Using Transactions

- **Keep transactions as short as possible**
 - Use caution with certain Transact-SQL statements
 - Avoid transactions that require user interaction
 - Do not browse data during a transaction
 - Affect the least rows possible with DML statements
 - Access the least rows possible with SELECT statements
- **Issues with nested transactions**
 - Allowed, but not recommended
 - Use @@trancount to determine nesting level

Restricted Statements

- **Certain statements may not be included in explicit transactions, such as:**

• ALTER DATABASE	• RECONFIGURE
• BACKUP	• RESTORE DATABASE
• CREATE DATABASE	• RESTORE
• DROP DATABASE	• UPDATE STATISTICS
- **Full-text system stored procedure calls may not be included in explicit transactions**
- **You cannot use the following in implicit or explicit transactions:**
 - sp_dboption
 - System stored procedures that modify master

SQL SERVER Locking Architecture

What Concurrency Problems Are Prevented by Locks?

- Lost updates
- Uncommitted dependencies (dirty read)
- Inconsistent analysis (nonrepeatable read)
- Phantom reads

Lockable Resources

Item	Description
RID	Row identifier
KEY	Row lock within an index
PAGE	Data page or index page
EXTENT	Group of pages
TABLE	Entire table
HOB	A heap or B-tree
FILE	A database file
APPLICATION	An application-specified resource
METADATA	Metadata locks
ALLOCATION_UNIT	An allocation unit
DATABASE	Entire database

Types of Locks

- **Basic locks**
 - Shared
 - Exclusive
- **Special situation locks**
 - Intent
 - Update
 - Schema
 - Bulk update

Lock Compatibility

- **Some locks are compatible with other locks, and some locks are not**
- **Examples**
 - Shared locks are compatible with all locks except exclusive
 - Exclusive locks are not compatible with any other locks
 - Update locks are compatible only with shared locks

An example

Pisa Tours Agency Reservation System



Lost update

reservations

Seat	Name	...
7C	Steven	
7B	_____	
...		

App A

```
UPDATE reservations
SET Name = 'Susan'
WHERE Seat = '7C'
```

App B

```
UPDATE reservations
SET Name = 'Steven'
WHERE Seat = '7C'
```

26

Uncommitted read (also known as “dirty read”)

reservations

Seat	Name	...
7C	_____	
7B	_____	
...		

App A

```
UPDATE reservations
SET Name = 'John'
WHERE Seat = '7C'
```

ROLLBACK

App B

```
SELECT Name
FROM reservations
WHERE Seat = '7C'
```

John

Further processing in App B uses incorrect /uncommitted value of „John”

27

Non-repeatable read

reservations

Seat	Name	...
7C	John	
7B	_____	
...		

App A

```
SELECT Seat
FROM reservations
WHERE Name ISNULL
```

```
SELECT Seat
FROM reservations
WHERE Name ISNULL
```

7C

7B

7B

App B

```
UPDATE reservations
SET Name="John"
WHERE Seat = "7C"
```

The same SELECT (read) returns a different result: Less rows (in this case '7C' doesn't show anymore). This is a non-repeatable read.

28

Phantom read

reservations

Seat	Name	...
7C	_____	
7B	_____	
...		

App A

App B

7B

```
SELECT Seat
FROM reservations
WHERE Name ISNULL
```

7B

```
SELECT Seat
FROM reservations
WHERE Name ISNULL
```

7C

```
UPDATE reservations
SET Name=NULL WHERE
Seat = "7C"
```

The same SELECT (read) returns a different result: More rows (phantom rows, in this case '7C', is shown)
This is a phantom read.

Isolation levels

- “Policies” to control when locks are taken
- Database provides different levels of protection to isolate data
- Transaction isolation level
 - Read Uncommitted
 - Read Committed (default)
 - Repeatable Read
 - Serializable
- Locking timeout
 - Limits time waiting for a locked resource
 - Use SET LOCK_TIMEOUT

weak

strong

Isolation levels and Concurrency Problems

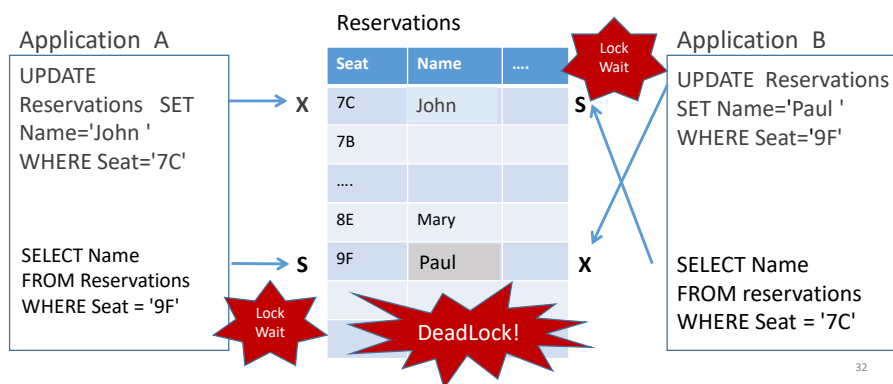


	Dirty reads	Non-repeatable read	Phantoms
Read Uncommitted	Yes	Yes	Yes
Read Committed	No	Yes	Yes
Repeatable read	No	No	Yes
Serializable	No	No	No

31

Deadlocks

- Occurs when two or more applications wait indefinitely for a resource
- Each application is holding a resource that the other needs
- Waiting is never resolved



32

Deadlocks

Database servers :

- provide a deadlock detector
- set the time interval for checking for deadlocks
- When a deadlock is detected, database server uses an internal algorithm to pick which transaction to roll back, and which one to continue.
- The transaction that is forced to roll back gets a SQL error. The rollback causes all of its locks to be released.

33

Special Type of data

34

Geometry and Geography data types

- Geometry data type
 - The GEOMETRY data type might be used for a warehouse application to store the location of each product in the warehouse
 - The GEOMETRY data type follows a “flat Earth” model, with basically X, Y, and Z coordinates.
- Geography data type
 - The GEOGRAPHY data type can be used to store data that can be used in mapping software. You may wonder why two types that both store locations exist. The GEOGRAPHY data type represents the “round Earth” model, storing longitude and latitude. These data types implement international standards for spatial data.
- These data types supports the OpenGIS Simple Features for SQL standard, which is a specification published by an international regulatory body known as the Open Geospatial Consortium (OGC):
 - Well-Known Text (WKT),
 - Well-Known Binary (WKB),
 - Geography Markup Language (GML).












35

- Shapes are projected onto spatial models using vector objects—which are collections of points, lines, and polygons (closed shapes). Both the geometry and geography data types support the same Well-Known Text (WKT) markup language, which is a convention that expresses the vector objects that you define using a syntax governed by the OGC.

Examples of WKT Strings	
WKT String	Description
POINT(6 10)	A single point at xy-coordinates 6, 10
POINT(-111.06687 45.01188)	A single point on the earth (longitude/latitude coordinates)
LINESTRING(3 4,10 50,20 25)	A two-part line, drawn between three points specified as xy-coordinates
POLYGON((-75.17031 39.95601, -75.16786 39.95778, -75.17921 39.96874, -75.18441 39.96512, -75.17031 39.95601))	An enclosed shape on the earth drawn between the points specified as longitude/latitude coordinates

As you can see, the same WKT syntax is used for expressing spatial elements using either the planar or the geodetic model. Also notice that geodetic coordinates are always expressed in WKT with the longitude value first, followed by the latitude value.

Geometry data type

POINT		MULTIPOINT	
LINESTRING		MULTILINESTRING	
CIRCULARSTRING		COMPOUNDCURVE	
POLYGON		MULTIPOLYGON	
CURVEPOLYGON		GEOMETRY-COLLECTION	
FULLGLOBE			

37

Geometry and Geography data types

- These complex types are based on the CLR (Common Language Runtime). New data types similar to these built-in complex types can be created with a .NET language. We should know how to use the built-in CLR types.
- These types are in .NET as a class library (DLL) and offer us more than 90 methods. The name of these methods begin with the ST characters symbolize that support the OGC standard.
- Some of the methods are static and some of them are object instance methods.

38

Geometry data type

- Planar model is a flat surface where shapes are plotted using two-dimensional x- and y-coordinates
- These coordinates are based on an arbitrary measurement system, so you can define any measurement unit you want (for example, centimeters, meters, kilometers, inches, feet, miles, pixels, and so on).
- Our first example demonstrates the geometry data type in a very simple scenario. You will define and store shapes representing different objects.
- The first thing you need to do is create tables to hold the shapes that define the warehouse and objects to place in it.
- You can give the area, and distance between them.

```
CREATE TABLE Warehouse(ObjectName nvarchar(20),
                        Place GEOMETRY);
```

39

Geometry datatype

- Let be walls of the warehouse:

```
INSERT Warehouse VALUES ('Walls', 'LINESTRING(0 0, 40 0, 40 40, 0 40, 0 0)')
```

- Put some object into it.

```
INSERT Warehouse VALUES('Bean','POINT(5 35)');
INSERT Warehouse VALUES('Rod','LINESTRING(10 10, 25 25)');
INSERT Warehouse VALUES('Table','POLYGON((15 18, 35 18, 35 28, 15 28, 15 18))');
INSERT Warehouse VALUES('Swimbelt','CURVEPOLYGON(CIRCULARSTRING(0 4,4 0,8 4,4 8,0 4),
CIRCULARSTRING(2 4,4 2,6 4,4 6,2 4))');
```

40

- Calculate the area occupied by objects:

```
SELECT *, Place.STArea() as Area
FROM Warehouse;
```

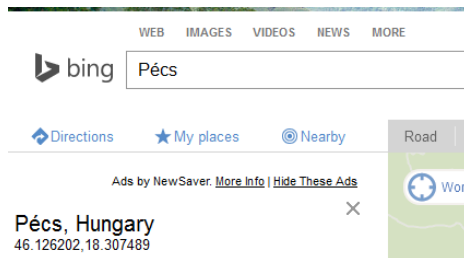
[illegible]

Geography datatype

- Create City table with location

```
-- Create CITY table
CREATE TABLE CITY(CityID INTEGER IDENTITY(1,1) PRIMARY KEY NOT NULL,
  CityName nvarchar(12), CityLoc Geography);
GO
```

- Add data to the City table



43

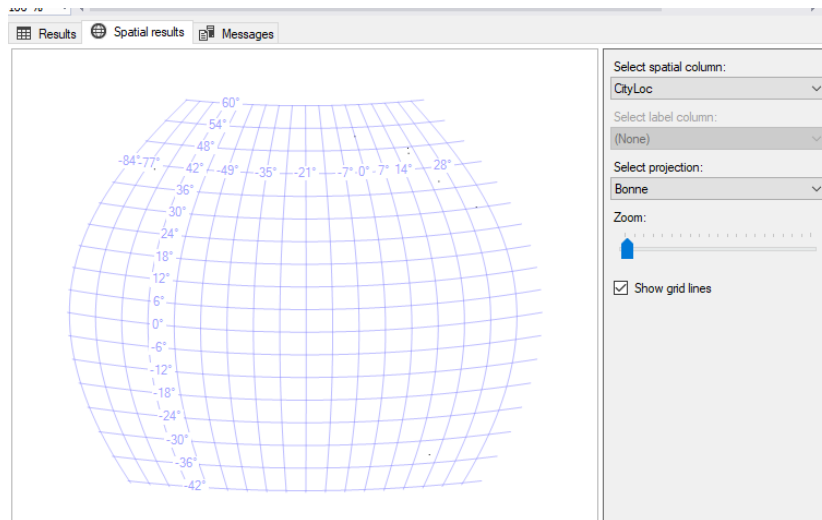
Geography datatype

- Insert data into the City table:

```
-- INSERT Data into the City table, Longitude, Latitude
INSERT City(CityName, CityLoc)
VALUES('Budapest', 'POINT(19.064819 47.506221)')
INSERT City(CityName, CityLoc)
VALUES('Pécs', 'POINT(18.307489 46.126202)')
INSERT City(CityName, CityLoc)
VALUES('London', 'POINT(-0.127140 51.506321)')
INSERT City(CityName, CityLoc)
VALUES('Athens', 'POINT(23.736410 37.976150)')
INSERT City(CityName, CityLoc)
VALUES('New York', 'POINT(-74.007118 40.714550)')
INSERT City(CityName, CityLoc)
VALUES('Cape Town', 'POINT(18.421989 -33.919090)')
INSERT City(CityName, CityLoc)
VALUES('Cairo', 'POINT(31.235711 30.0444196)')
GO
```

44

The result



The distance between two city is calculated with `STDISTANCE()` function

45

The distances between cities

```
SELECT
  C1.CityName AS City1, C2.CityName AS City2,
  ROUND(C1.CityLoc.STDistance(C2.CityLoc)/1000,2) AS Km
FROM City AS C1 JOIN City AS C2 ON C1.CityID<C2.CityID
ORDER BY C1.CityID;
```

	City1	City2	Km
1	Budapest	Pécs	163.94
2	Budapest	London	1454.51
3	Budapest	Athens	1125.11
4	Budapest	New York	7027.36
5	Budapest	Cape Town	9018.44
6	Budapest	Cairo	2202.07
7	Pécs	London	1474.88
8	Pécs	Athens	1010.07
9	Pécs	New York	7059.6
10	Pécs	Cape Town	8864.81
11	Pécs	Cairo	2108.71
12	London	Athens	2396.11
13	London	New York	5585.27
14	London	Cape Town	9635.63
15	London	Cairo	3513.89
16	Athens	New York	7945.69
17	Athens	Cape Town	7978.51
18	Athens	Cairo	1118.89
19	New York	Cape Town	12552.02
20	New York	Cairo	9040.98
21	Cape Town	Cairo	7206.9

46

Thank you for your attention!