


OLTP

- Online Transaction Processing 
- Transaction-oriented Applications
- Mostly connected to applications/software, designed towards business needs / logic
- Daily Uses
- Contains Active Data
- May contains some historical data, based on business needs
- Usually structured data combined with relational databases and nosql databases

ODS



- Operational Data Store
- A “merge” area for data coming from different sources
- Great place to do data cleansing, reporting and staging
- Usually contains active data for a desired period of time (7-60 days)
- Data comes in and out by batches (minutes to hours)
- Some enterprise solutions would have a separate staging database between ODS and DW.
- Usually a somewhat normalized relational structure
- structure will be in between OLTP and OLAP

DW/OLAP



- Data Warehouse
- Contains lots of historical data
- Data coming from ODS or Staging in batches, often never deleted
- Data used for analytic and reports only
- Usually snowflake schema or star schema. Not normalized to save joining time
- Usually accessible through TSQL and other server built in languages

Data Lake

- Similar to Data Warehouse, lots of Historical data for analytics, data quality is hard to guarantee
- Can store structured, semi-structured and unstructured data(basically anything encoded in binary)
- little to no constraints on data going in
- when poorly managed, could easily turn into a “data swamp”
- typically easy to get data into, but hard to get insights out.

Normalization

- Improves data integrity -> PK/FK
- Decrease storage costs
- Increase Join time
- 1st Normal Form
 - each cell in the table should contain only atomic values
- 2nd Normal Form:
 - every non candidate-key attribute must depend on the whole candidate key, not just part of it
- 3rd Normal Form:
 - remove transitive dependency, non-key attributes can not depend on other non-key attributes.

Relational DB Design

- Focus on normalization and relationships
- Entity Table
 - represents business entities
 - book, user, student, courses....
- Lookup Table
 - enumerate/list all potential cases, mostly to supplement entity tables
- Joint/junction Table
 - used to handle many-to-many relationships

Relational DB Design

- Identity Column
 - seed specifies the starting value
- Audit column
 - status, create date, modified date, created by, modified by
- Computed/Derived Column
 - Computed from other columns, not stored, computed when needed
- GUID/UUID
- Natural vs Surrogate keys
- PII/Sensitive data(data masking)

```
IDENTITY [ (seed , increment) ]
```

```
CREATE TABLE new_employees  
(  
    id_num int IDENTITY(1,1),  
    fname varchar (20),  
    minit char(1),  
    lname varchar(30)  
);
```

```
CREATE TABLE dbo.Products  
(  
    ProductID int IDENTITY (1,1) NOT NULL  
    , QtyAvailable smallint  
    , UnitPrice money  
    , InventoryValue AS QtyAvailable * UnitPrice  
);
```

Data Warehouse Design

- focus on star/snowflake and Fact/Dimension
- SCD(slowly changing dimensions) - type 2

Performance Monitoring

- Locate issue on a **server** scale
 - **Extended Events**
 - SQL Server Profiler - performance hit
 - DMV
 - DBCC(Database Console Command)
- Locate issue on a **workload** scale
 - **Tuning Advisor**
 - **Execution Plan**
- <https://docs.microsoft.com/en-us/sql/relational-databases/performance/performance-monitoring-and-tuning-tools?view=sql-server-ver15>

Performance Tuning

- Unnecessary group by
 - reduce the columns in groupby
- Correlated subquery
 - subquery needs to execute for every row of outer query
- Join condition vs Where condition
 - filter data before the join (with join condition)
- Temp table, table variable
 - when data volume is big, consider temp table +index
- Be smart with Locks, Isolation Levels
- Missing indexes
- Bulk insert or massive updating
 - might block other operations
- Recompile of SP, Functions
 - avoid parameter sniffing related issues

```
ALTER PROCEDURE Get_OrderID_OrderQty
@ProductID INT
WITH RECOMPILE
AS

SELECT SalesOrderDetailID, OrderQty
FROM Sales.SalesOrderDetail
WHERE ProductID = @ProductID
```