Data Warehousing

COSC 304 – Introduction to Database Systems

OLAP and OLTP

Databases designed to handle many small queries and updates are referred to as ***online transaction processing*** (***OLTP***) systems.

Online analytical processing (***OLAP***) systems are designed for decision support applications where large amounts of data is analyzed often with *ad hoc* queries.

Comparison of OLTP versus OLAP:

OLTP	OLAP
For transaction data	For historical data
Data is dynamic	Data is static
Common transactions	Ad hoc and varying queries
Transaction driven	Analysis driven
Large number of users	Smaller number of users

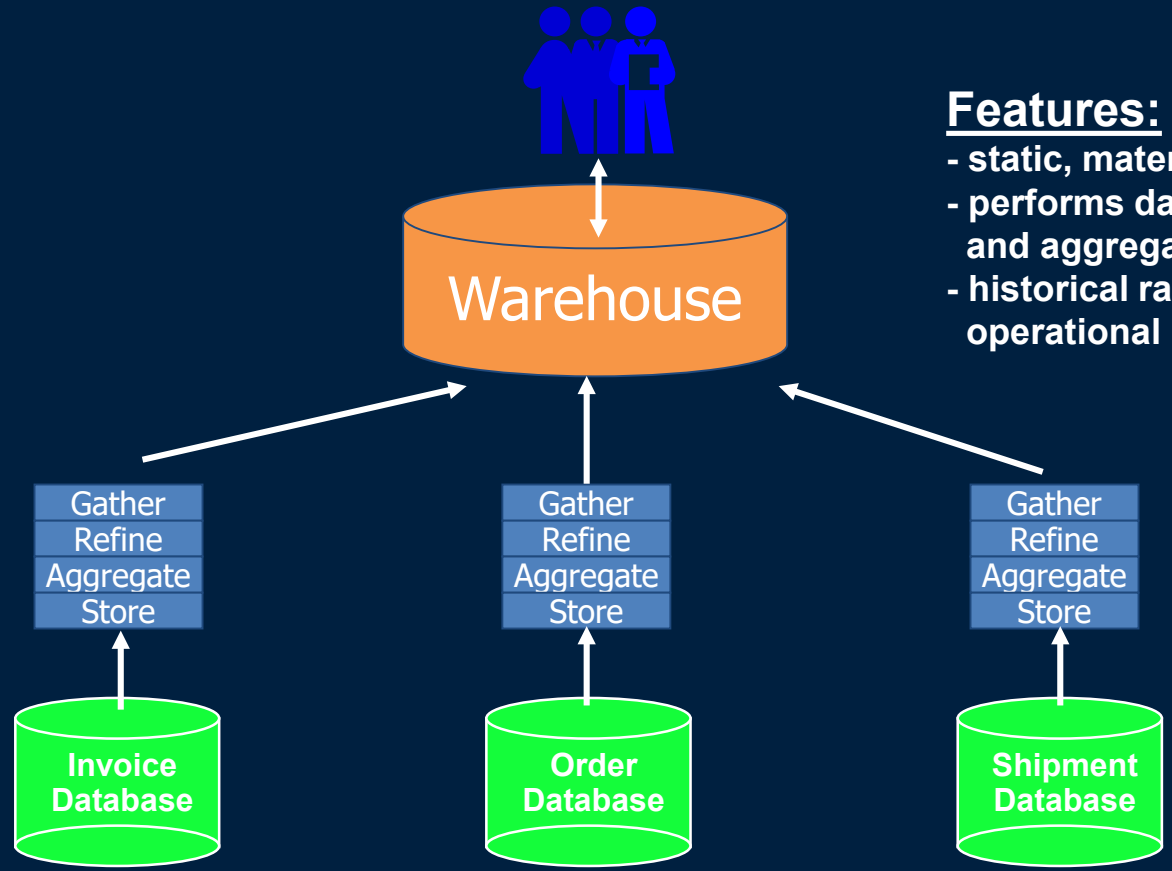
Data Warehousing

It is not practical to have long, analytical queries running on operational systems. One solution is to create a separate database that contains a copy of the operational data that is summarized and organized in an efficient manner.

A ***data warehouse*** is a historical database that summarizes, integrates, and organizes data from one or more operational databases in a format that is more efficient for analytical queries.

A ***data mart*** is a subset of a data warehouse that supports the business requirements of one department.

Data Warehouse Approach



Features:

- static, materialized view
- performs data cleansing and aggregation
- historical rather than operational

Building a Data Warehouse

The general steps to building a data warehouse are:

- 1) Identify the data available in operational systems that is necessary for analytical queries.
- 2) Build a model for the data warehouse, typically as a star schema.
- 3) Write extraction and cleansing routines (often using design tools) for exporting data from operational systems and importing it into the data warehouse.
  - Data warehouses are often refreshed nightly.
- 4) Develop pre-made analysis reports and queries on the data warehouse and provide access to ad hoc query facility.
- 5) Formalize procedures for maintenance of the data warehouse and associated metadata including backup, refresh procedure, user access, and how changes to operational systems will be supported.

Data Warehouse MetaData

Metadata is data about data, or schema information.

In a data warehouse, metadata not only includes the type and name of fields, but also the source of the data, how and when it was extracted, and what transformations were applied to it before it was loaded into the warehouse.

Data warehouse metadata provides the history of every item in the warehouse. Maintaining accurate metadata is a critical documentation challenge.

Data Warehouse Tools

There are tools available for supporting the construction of data warehouses. Using these tools is critical to the successful and timely deployment of a data warehouse.

The types of tools include:

- Extraction and Cleansing Tools
  - Automate common tasks of extracting data from a data source, applying transformations, and then loading data into another data source.
- Report Design Tools
  - For the construction of common reports.
- Analytical (OLAP) User Tools
  - For supporting ad hoc querying to the data warehouse.
- Administration and Management Tools
  - For backup and monitoring data warehouse performance.

Star Schemas

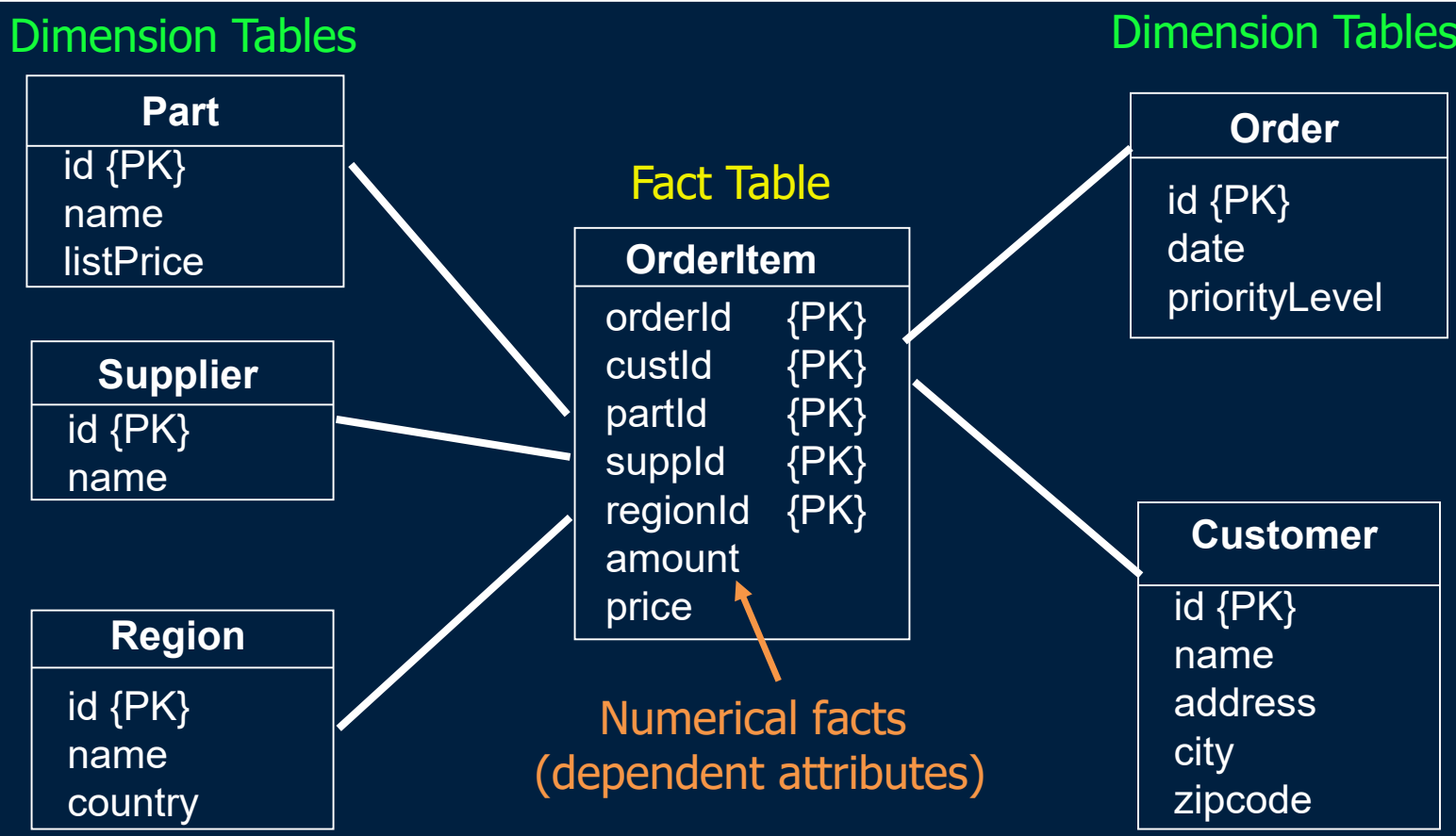
Data warehouses are designed using a technique called ***dimensionality modeling*** where:

- A main table exists called a ***fact table*** with a composite primary key and foreign keys to one or more ***dimension tables***.
- A fact table consists of:
  - *Dimension attributes*: key attributes of a dimension table.
  - *Dependent attributes* : attributes determined by the dimension attributes

When drawn in a diagram, the schema looks like a star because the fact table is in the middle with multiple dimension tables linked to it. Hence, schemas for data warehouses are often called ***star schemas***.

The fact table is the largest table in the data warehouse and grows the fastest. All tables are treated as read-only.

Example Star Schema



OLAP Queries

A typical OLAP query on a star schema involves:

- a star join connecting the dimensions to the fact table
- selection of the data using the dimensions
- group by one or more dimensions
- aggregate the numeric facts/values

Example: For each customer in the city "New York", find the total number of products ordered from supplier "ABC":

```
SELECT     C.name, SUM(amount)
FROM       OrderItem OI, Customer C, Supplier S
WHERE      OI.custId = C.id and OI.suppId = S.id
             and S.name = 'ABC' and C.city = 'New York'
GROUP BY   C.id, C.name;
```

Other Design Issues

For efficiency and simplicity, natural keys such as a SSN or student # are replaced with autoincrement (integer) fields.

- This is done to save space and reduce the number of changes necessary if a key field happens to change. Such keys are called ***surrogate keys***.

Data Warehouse Design Question

We have been using the following example WorksOn database:

emp (eno, ename, bdate, title, salary, supereno, dno)
proj (pno, pname, budget, dno)
dept (dno, dname, mgreno)
workson (eno, pno, resp, hours)

Assume the database stores the number of hours worked on a weekly basis. Each week the tuples in the WorksOn relation are deleted. Weeks are numbered by the company as 1,2,..,52.

Design a star schema for this data.

Efficient Processing of OLAP Queries

There are some techniques that can speed the processing of OLAP queries.

Materialized views are views that are physically stored in the database. Materialized views speed up OLAP queries by pre-computing large joins.

Bitmap indexes allow for efficient lookup of data values. In bitmap indexing, a bit vector is created for each possible value of the attribute being indexed. The length of the bit vector is the number of tuples in the indexed table. The *j*-th bit of the vector is 1 if tuple *j* contains that value.

- Obviously, this is most effective when the domain of the attribute is small as a bit vector is needed for each value.

Materialized View Example

Consider the query, for each customer in the city "New York", find the total number of products ordered from supplier "ABC":

```
SELECT     C.name, SUM(amount)
FROM       OrderItem OI, Customer C, Supplier S
WHERE      OI.custId = C.id and OI.suppId = S.id
           and S.name = 'ABC' and C.city = 'New York'
GROUP BY   C.id, C.name;
```

Useful materialized view:

```
CREATE VIEW v1(custId, custName, suppName, amount) AS
SELECT     C.id, C.name, S.name, SUM(amount)
FROM       OrderItem OI, Customer C, Supplier S
WHERE      OI.custId = C.id and OI.suppId = S.id
GROUP BY   C.id, C.name, S.id, S.name;
```

Materialized View Example (2)

This view can be used to simplify the query to:

```
SELECT      C.name, SUM(v1.amount)
FROM        v1, Customer C
WHERE       v1.custId = C.id and v1.suppName = 'ABC'
            and C.city = 'New York'
GROUP BY    C.id, C.name;
```

A view may significantly increase performance as the large join and aggregation involving the fact table is pre-computed.

Materialized views are especially effective in data warehouses because they are easy to maintain as the data is rarely changed except during loading.

- It is an active research area to determine what is the best set of views to materialize given a set of queries to answer.

ROLAP versus MOLAP

There are two approaches to building a data warehouse:

- ***ROLAP*** - relational OLAP that uses a standard relational DBMS and star schema to store warehouse data.
- ***MOLAP*** - multidimensional OLAP that uses a specialized DBMS with a data cube model.

The advantage of ROLAP is that it can use existing DBMS technology. However, MOLAP systems can be more readily tuned for dimensional data.

MOLAP and Data Cubes

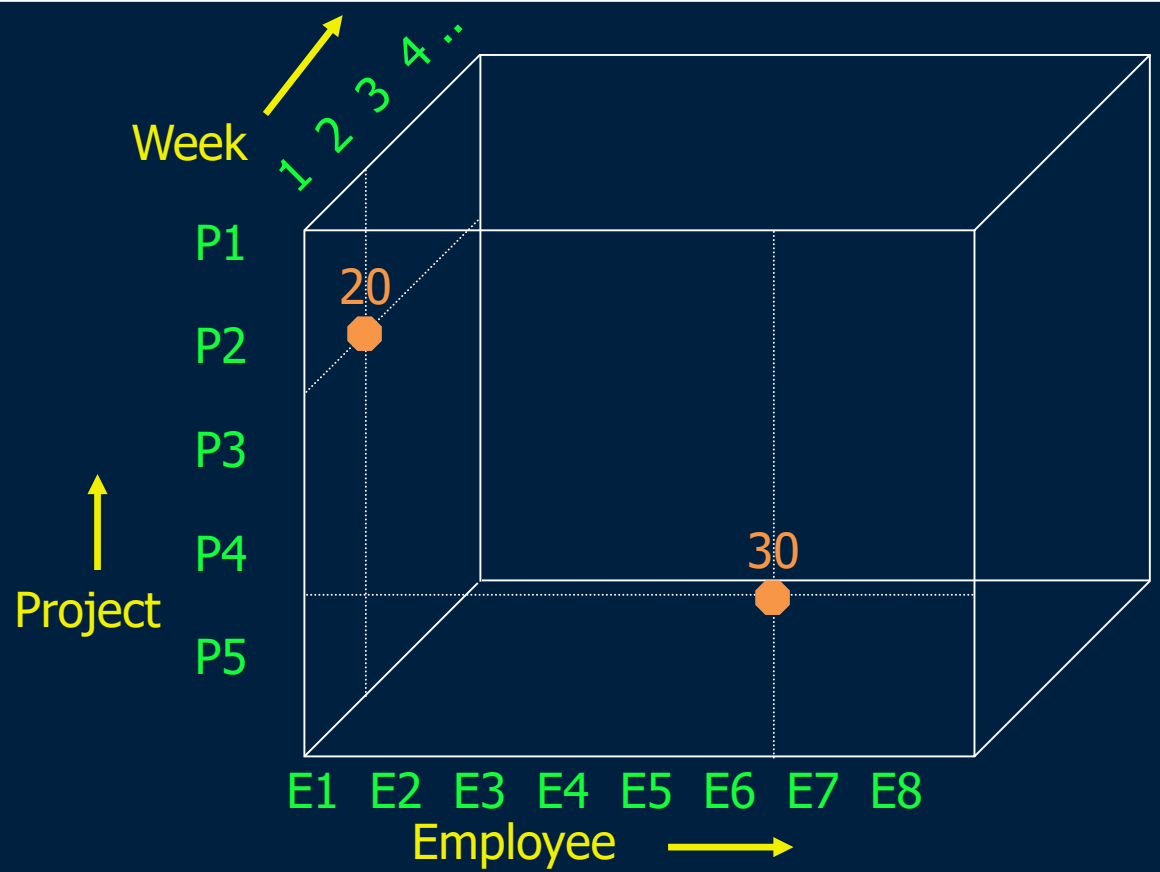
The data structure in MOLAP is a ***data cube*** or ***hypercube***. A data cube can be thought of as a *N*-dimensional spreadsheet.

The keys of the dimension tables are the dimensions of the cube. Dependent attributes appear at the points of the cube. Thus, a tuple consists of the:

- key attributes - defines the position of the point in the cube
- dependent attributes - one or more values at that position in the cube

The data cube also includes aggregation along the margins of the cube. These ***marginals*** may include aggregations over one or more dimensions.

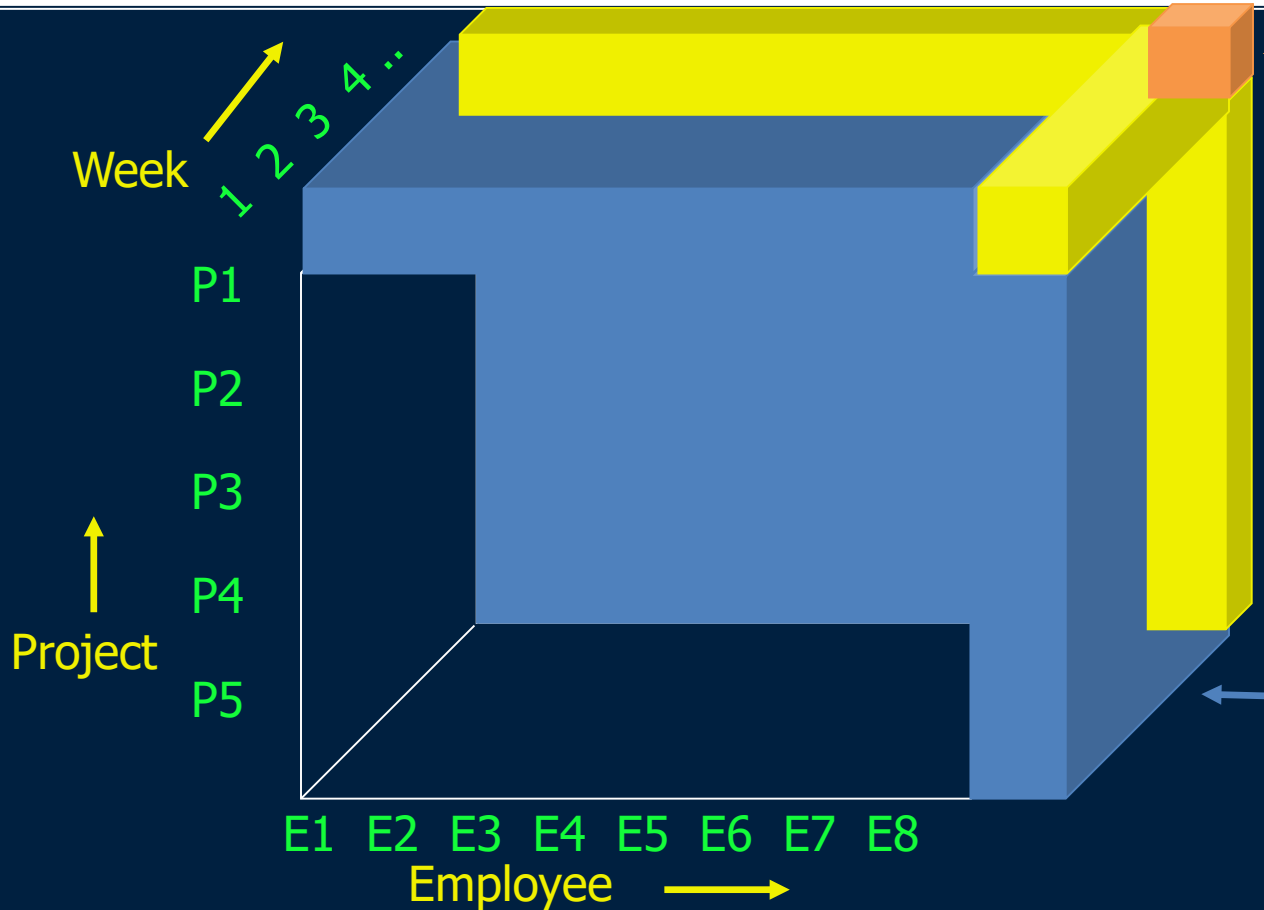
Data Cube Example



Fact #1: Worked (E1, P2, 2, 20)

Fact #2: Worked (E6, P4, 1, 30)

Data Cube Example Marginals



Sum of all data points in all 3 dimensions
- Worked(*,*,*)

Summary in 2 dimensions
- Top/Back: Worked(E,*,*)
- Top/Right: Worked(*,*,W)
- Back/Right: Worked(*,P,*)

Summary in 1 dimension
- Left: Worked(*,P,W)
- Back: Worked(E,P,*)
- Top: Worked(E,*,W)

Cube Aggregations

Think of each cube dimension as having an additional value * which stands for all values. A point with one or more *'s aggregates over the dimensions with the *'s.

Examples:

- `Worked("E1",*,*)` = sum of all hours `E1` has ever worked
- `Worked("E2","P3",*)` = sum of hours `E2` has worked on `P3`
- `Worked(*,*,*)` = sum of hours worked by all employees on all projects

Question: How would you answer these queries using the marginals in the data cube?

Cube Operations

Slice performs a select on one or more dimensions.

- Slice corresponds to adding a `WHERE` clause in SQL.

Dice is used to partition or group on one or more dimensions

- Dice involves dividing a cube into smaller sub-cubes.
- Dice corresponds to adding a `GROUP BY` clause in SQL.

Cube Navigation

Data analysts often "navigate" a data cube by summarizing data (roll-up) to get more general information or drilling-down to get more specific information.

Drill-down is the process of dividing an aggregated value into its component parts.

- Example: `Worked("E1",*,*)` shows that `E1` has not worked too many hours. Can drill-down by project (2nd `*`) to show how many hours `E1` has worked per project.

Roll-up is the process of aggregating along a particular dimension (opposite of drill-down).

- Example: `Worked("E1","P1",1)` shows `E1` has worked 20 hours in week1. Can roll-up on week: `Worked("E1","P1",*)` to get total hours `E1` has worked on `P1` over all weeks.

What is Integration?

Two levels of integration:

- ***Schema integration*** is the process of combining local schemas into a global, integrated view by resolving conflicts present between the schemas.
- ***Data integration*** is the process of combining data at the entity-level. It requires resolving representational conflicts and determining equivalent keys.

Integration handles the different mechanisms for storing data (structural conflicts), for referencing data (naming conflicts), and for attributing meaning to the data (semantic conflicts).

Integration Problems

Integration problems include:

- Different data models and conflicts within a model
- Incompatible concept representations
- Different user or view perspectives
- Naming conflicts (homonym, synonym)

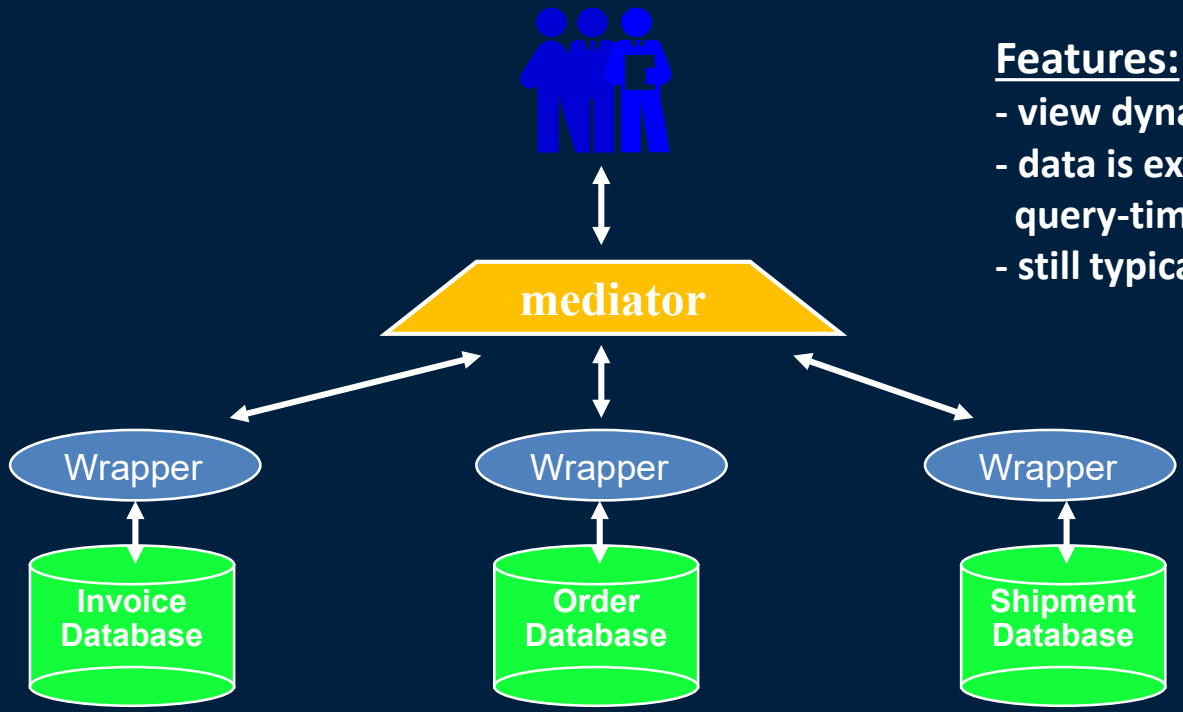
Integration using Mediators

Unlike integration using a data warehouse, integration architectures that use wrappers and mediators provide online access to operational systems.

Wrappers are software that converts global level queries into queries that the local database can handle. A ***mediator*** is global-level software that receives global queries and divides them into subqueries for execution by wrappers.

Unlike data warehouses, these systems are not suitable for decision-support queries because the data must be dynamically extracted from operational systems. They are useful for integrating operational systems without creating a single, unified database.

Query-Driven Dynamic Approach



Features:

- view dynamically built
- data is extracted at query-time
- still typically read-only

Integration Challenges

Database integration is an active area of research. Common problems include:

- 1) **Schema matching and merging** - How can we create a single, global schema for users to query? Can this be done automatically?
- 2) **Global Query Optimization** - How do we optimize the execution of queries over independent data sources?
- 3) **Global Transactions and Updates** - Is it possible to efficiently support transactions over autonomous databases?
- 4) **Global Query Languages** - Is SQL a suitable query language when the user does not understand the entire schema being queried?
- 5) **Peer-to-Peer** - What integration technologies are suitable for massive scale integrations over a grid or in dynamic peer-to-peer systems?

Conclusion

A ***data warehouse*** is a historical database that summarizes, integrates, and organizes data from one or more operational databases in a format that is more efficient for analytical queries.

- OLAP differs from OLTP as it consists of longer, ad hoc queries that access more of the data in the database.

Building a data warehouse involves identifying operational sources, extracting and cleansing data, and loading the data into the warehouse.

Star schemas are used when a data warehouse is modeled in a relational DBMS. MOLAP stores the data in a cube and has operators such as slice, dice, roll-up, and drill-down.

Objectives

- List some differences between OLTP and OLAP.
- Define data warehouse and describe some key features.
- Describe the steps in building a data warehouse.
- Explain how metadata is used in a data warehouse.
- Given a star schema, identify the fact table, dimension table, dimension attributes, and dependent attributes.
- List two ways the relational DBs speed-up processing OLAP queries (materialized view and bitmap indexes).
- Explain why materialized views are beneficial for warehouses.
- Compare the advantages of ROLAP versus MOLAP.
- Define the 2 cube operators: slice and dice.
- Explain cube navigation using drill-down and roll-up.
- Compare integration using mediators versus data warehousing.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA