22.1 Cloud databases

Multi-tier architecture

Multiple computers linked by a network are often grouped in layers, called ***tiers***, and arranged in a hierarchy.

Prior to 1990, most software ran in a ***single-tier architecture***, consisting of a personal or corporate computer connected directly to monitors. Although computers often communicated with each other, the dependencies between applications running on different computers were limited.

Since 1990, complex corporate and government applications have increasingly been implemented in a ***multi-tier architecture***:

- The top tier consists of computers interacting directly with end-users.
- The bottom tier consists of servers managing resources like databases and email.
- One or more middle tiers execute a variety of functions, such as user authorization, business logic, and communication with other computers.

Typically, application programs run on a middle tier and implement business logic. Since user interaction and data are managed in the top and bottom tiers, applications are easier to write and maintain in a multi-tier architecture.

Web architecture is a multi-tier architecture consisting of web browsers and web servers communicating over the internet:

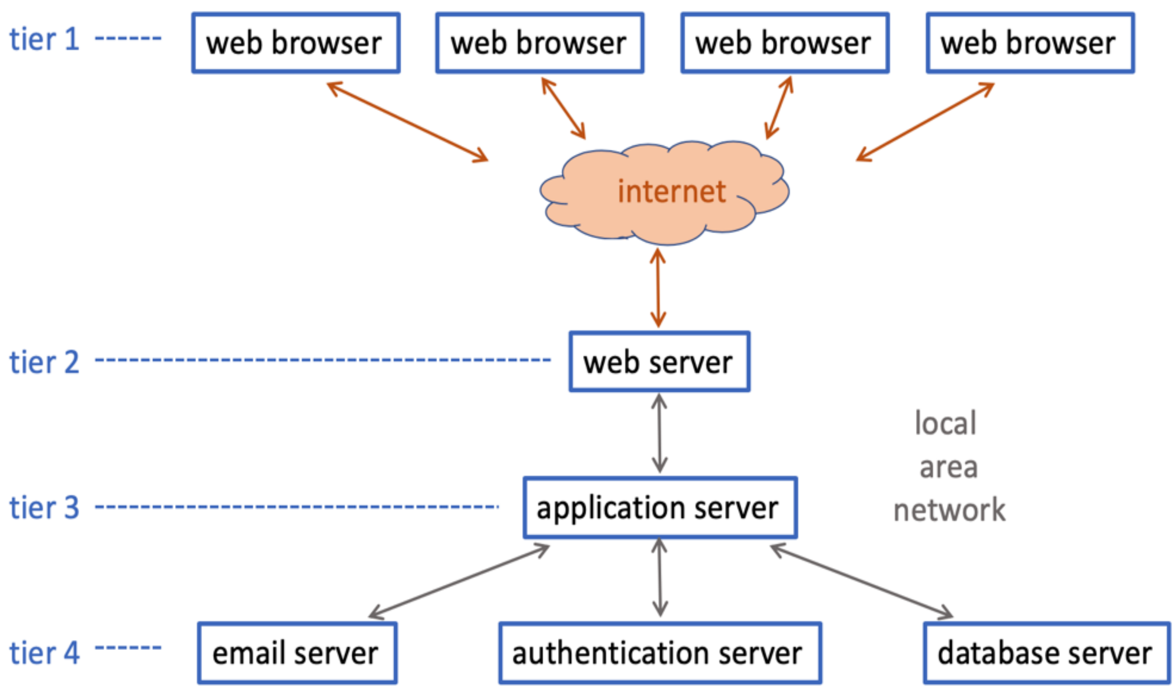
- Web browsers, on the top tier, manage user interaction.
- Web servers, on a middle tier, generate web pages for display on web browsers and transmit user requests to services running on lower tiers.
- Application servers run application software, process user requests, and communicate with databases and other services.
- Services, such as database and authentication, comprise the bottom tier.

Web architecture proliferated as internet use grew rapidly during the 1990s and is now a dominant architecture.

Terminology

*The term **tier** refers to either a software or hardware layer. In this material, tier refers* *to a hardware layer.*

Figure 22.1.1: Web architecture uses multi-tier architecture.



PARTICIPATION ACTIVITY 22.1.1: Multi-tier architecture.

Match the architecture to the description.

If unable to drag and drop, refresh the page.

Multi-tier **Single-tier** **Web**

Contains two or more tiers. Tiers are connected by any communication technology, including local and wide area networks.

Typically contains many tiers, with

the top two tiers connected by the internet.

The dominant architecture prior to 1990.

Reset

Cloud databases

Prior to 2000, most commercial software was ***on-premise***, or installed and run on customer computers. Since 2000, cloud services have increasingly replaced on-premise software. With ***cloud services***, a vendor such as Amazon, Microsoft, or Google implements computer services on lower tiers of a web architecture. For a fee, cloud services are made available over the internet to customers.

Cloud services fall into three broad categories:

- ***Infrastructure-as-a-service***, or ***IaaS***, provides computer processing, memory, and storage media, as if the customer were renting a computer. Ex: Elastic Compute Cloud, or EC2, from Amazon Web Services offers infrastructure-as-a-service.
- ***Platform-as-a-service***, or ***PaaS***, provides tools and services, such as databases, application development tools, and messaging services. Ex: Azure is Microsoft's cloud services environment, offering the SQL Database service.
- ***Software-as-a-service***, or ***SaaS***, provides complete applications, usually through web browsers on customer machines. Ex: Salesforce offers sales management software, and Google offers document processing applications like Docs, Sheets, and Pages.

Usually cloud services are offered on virtual machines. A ***virtual machine***, or ***VM***, is a software layer that emulates a complete, independent computing environment. Multiple virtual machines can run on one computer, enabling cloud providers to support many customers on the same machine.

A ***cloud database*** is a database offered as a PaaS cloud service. Most databases are now available either on-premise or as a cloud service, but cloud database use is growing rapidly.

PARTICIPATION ACTIVITY | 22.1.2: Cloud services.

On-premise
IaaS
PaaS
SaaS
web browser

web browser	🟧	🟧	🟧	🟧
application	🟧	🟧	🟧	🟦
utilities and tools	🟧	🟧	🟦	🟦
database	🟧	🟧	🟦	🟦
operating system	🟧	🟧	🟦	🟦
virtual machine	🟧	🟦	🟦	🟦
server	🟧	🟦	🟦	🟦
network	🟧	🟦	🟦	🟦
storage media	🟧	🟦	🟦	🟦

🟧 customer 🟦 cloud provider

Animation content:

Static figure:
A grid appears with nine rows and four columns. The rows represent software and hardware layers. From top to bottom, the row labels are:
web browser
application
utilities and tools
database
operating system
virtual machine
server
network
storage media

The columns represent cloud service alternatives. The columns are labeled on-premise, IaaS, PaaS, and SaaS.

In the on-premise column, all rows are labeled customer. In the IaaS column, the web browser through operating system rows are labeled customer, and the remaining rows are labeled cloud provider. In the PaaS column, the web browser and application rows are labeled customer, and the remaining rows are labeled cloud provider. In the SaaS column, the web browser row is labeled customer, and the remaining rows are labeled cloud provider.

Step 1: Computer systems consist of multiple software layers. Row labels appear.

Step 2: With on-premise software, all layers are installed, administered, and run on customer computers. The on-premise column appears.

Step 3: With IaaS, cloud providers offer virtual machines to their customers. The IaaS column appears.

Step 4: With PaaS, cloud providers offer computer services and tools running within virtual machines. The IaaS column appears.

Step 5: With SaaS, cloud providers offer complete applications to customers. Usually, customers access applications via a web browser. The SaaS column appears.

Animation captions:

1. Computer systems consist of multiple software layers.
2. With on-premise software, all layers are installed, administered, and run on customer computers.
3. With IaaS, cloud providers offer virtual machines to their customers.
4. With PaaS, cloud providers offer computer services and tools running within virtual machines.
5. With SaaS, cloud providers offer complete applications to customers. Usually, customers access applications via a web browser.

Terminology

A premise is a statement or proposition, from which another statement is inferred. ***Premises*** *refers to buildings and land occupied by a business. Thus,* ***on-premises*** *is technically correct in the context of cloud software. However* ***on-premise*** *is easier to say and therefore commonly used.*

PARTICIPATION ACTIVITY 22.1.3: Cloud services.

1) 'Middleware' is a software layer between the operating system and applications, such as a web server or messaging software. Which category provides middleware in the cloud?

- ○ On-premise
- ○ IaaS
- ○ PaaS

2) Can virtual machines running on the same computer contain different operating systems?

- ○ Yes
- ○ No

3) How many tiers are contained in a SaaS environment?

- ○ 1
- ○ 3
- ○ Depends on cloud provider and application

4) In what form is Microsoft SQL Server available?

- ○ On-premise only
- ○ As a cloud service only
- ○ Either on-premise or as a cloud service

Benefits and risks

Cloud databases have a number of compelling benefits:

- *Administration*. Installing, managing, upgrading, and backing up database systems is time-consuming and complex. With cloud databases, consumers delegate administrative activities to cloud providers.
- *Security*. Cloud providers are large companies with extensive resources. Cloud providers can invest heavily in security professionals and infrastructure, providing better security than most

cloud customers.

- *Reliability*. Cloud providers provide redundant computing systems with little or no down-time.
- *Elasticity*. Many organizations struggle with daily, monthly, or seasonal fluctuations in processing workload. By averaging fluctuations over many customers, cloud providers provide flexible database resources on demand.
- *Capital cost*. Cloud providers absorb all initial, or capital, costs of computers and facilities. Capital cost is recovered by cloud service fees.

Cloud databases raise data privacy questions. Companies entrust data to cloud providers, which may store data on servers located in countries with different privacy regulations. Ex: The European Union has adopted comprehensive data privacy regulations. In the United States, data privacy is governed by limited regulations in specific areas, such as medical and financial. As a result, a European company may avoid servers located in the United States.

Data privacy is a concern primarily for sensitive data, such as financial and medical applications. For organizations that do not manage sensitive data, cloud databases offer convincing benefits and have been widely adopted.

Table 22.1.1: Cloud database benefits and risks.

	Benefits	Risks
Administration	Delegated to cloud provider	Not under direct control
Security	Usually more secure than on-premise	Data transmission over public internet
Reliability	High	
Elasticity	High	
Capital cost	Low	
Data privacy		Regulations based on server location

PARTICIPATION ACTIVITY 22.1.4: Cloud database benefits.

Which of the following are common benefits of cloud databases?

1) Availability

○ True

○ False

2) Operating cost

○ True

○ False

3) Confidentiality of data

○ True

○ False

4) Ability to handle peak processing loads

○ True

○ False

Exploring further:

- Cloud services

CHALLENGE ACTIVITY | 22.1.1: Cloud databases.

544874.3500394.qx3zqy7

Start

(a) A service provides a scalable number of virtual machines to serve as large computer clusters.

(b) Google offers a Git
collaboration tools for
Customers build applic

Which layers are managed by the customer? □ □

web browser
----
application

□ □

□	utilities and tools	□
□	database	□
□	operating system	□
□	virtual machine	□
□	server	□
□	network	□
□	storage media	□
What is the service type?	Pick	Pick

1

Check   Try again

22.2 Distributed databases

Parallel computers and clusters

A ***parallel computer*** consists of multiple processors managed by a single operating system instance. Parallel computers achieve faster processing speeds by processing multiple instructions concurrently.

Parallel computers fall into three categories:

- In a ***shared memory*** computer, processors share the same memory and storage media.
- In a ***shared storage*** computer, processors share storage media only. Each processor has private memory.
- In a ***shared nothing*** computer, processors share neither memory nor storage media.

Shared memory is optimal for parallel processing against a common data set in a single memory space. However, shared storage and shared nothing scale to more processors, since processors do not contend for the same memory.

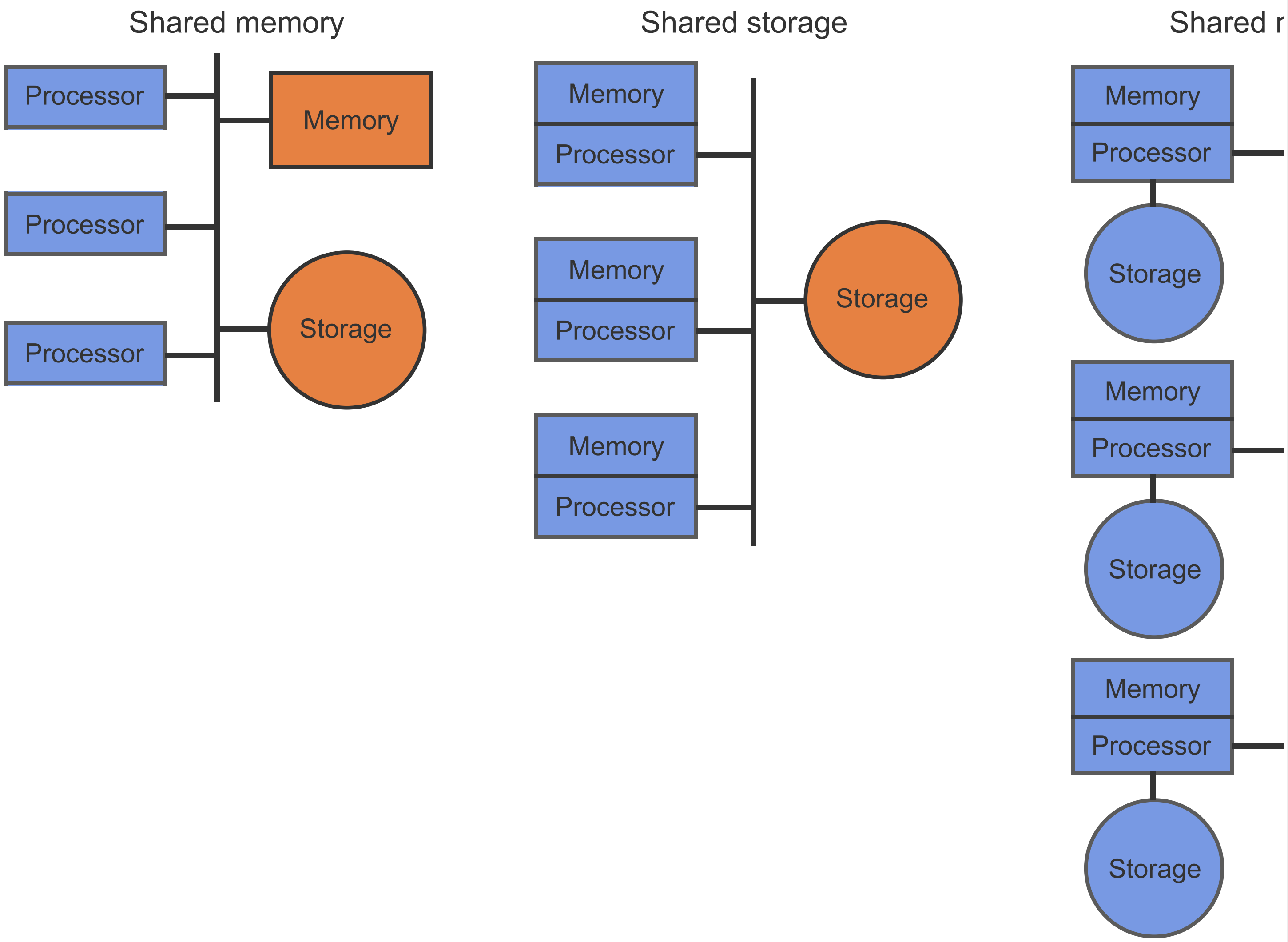
Multiple computers can communicate via a local or wide area network:

- A ***local area network*** consists of cables extending over a small area, typically within one facility. Local area networks usually use the Ethernet communication protocol.
- A ***wide area network*** spans multiple facilities in different geographic locations, separated by many miles. Wide area networks may communicate via cables, satellite, or telephone lines, often using internet communication protocols.

A ***node*** is one of a group of computers connected by either a local or wide area network. A ***cluster*** is a group of nodes connected by a local area network, managed by separate operating system instances, and coordinated by specialized cluster management software.

A cluster is similar to a parallel computer. Both can execute program instructions in parallel on multiple processors. Both can share storage or share nothing. Computers in a cluster cannot share memory, however, since local area networks are too slow to support memory access.

PARTICIPATION ACTIVITY | 22.2.1: Parallel computers.



Animation content:

Static figure:
Three diagrams appear, with captions shared memory, shared storage, and shared nothing.

Step 1: In a shared memory system, multiple processors share both memory and storage. The shared memory diagram has three processor icons, a memory icon, and a storage icon. All five icons are connected to a line that represents internal connections. Highlighting indicates memory and storage are shared.

Step 2: In a shared storage system, multiple processors share storage only. The shared storage diagram has three icons containing both processor and memory, and one storage icon. All four icons are connected to a line that represents internal connections. Highlighting indicates storage is shared.

Step 3: In a shared nothing system, each processor has private memory and storage. The shared nothing diagram has three icons containing processor, memory, and storage. All three icons are connected to a line that represents internal connections. Nothing is highlighted or shared.

Animation captions:

1. In a shared memory system, multiple processors share both memory and storage.
2. In a shared storage system, multiple processors share storage only.
3. In a shared nothing system, each processor has private memory and storage.

PARTICIPATION ACTIVITY | 22.2.2: Parallel computers and clusters.

Indicate whether each of the following configurations is shared memory, shared storage, or shared nothing.

1) Four computers in one building are connected by cables. The computers connect to a bank of disk drives, also on the network, via the Ethernet protocol.

○ Shared memory

○ Shared storage

○ Shared nothing

○ Shared nothing

2) A parallel computer has one CPU containing a dozen processors, called 'cores'. The CPU and main memory are in separate integrated circuits on one circuit board. The circuit board connects to a flash drive via an internal communications 'bus'.

○ Shared memory

○ Shared storage

○ Shared nothing

3) Two computers in different cities communicate via the internet. Each computer has a local disk drive. Each computer can access the remote disk drive indirectly by communicating with the other computer.

○ Shared memory

○ Shared storage

○ Shared nothing

Parallel and distributed databases

Queries can often be decomposed into parts that run concurrently and execute faster on parallel computers or clusters. Ex: If a query joins tables stored on different disk drives, different processors can read and sort each table in parallel.

Parallel and distributed databases exploit multiple processors for faster query execution:

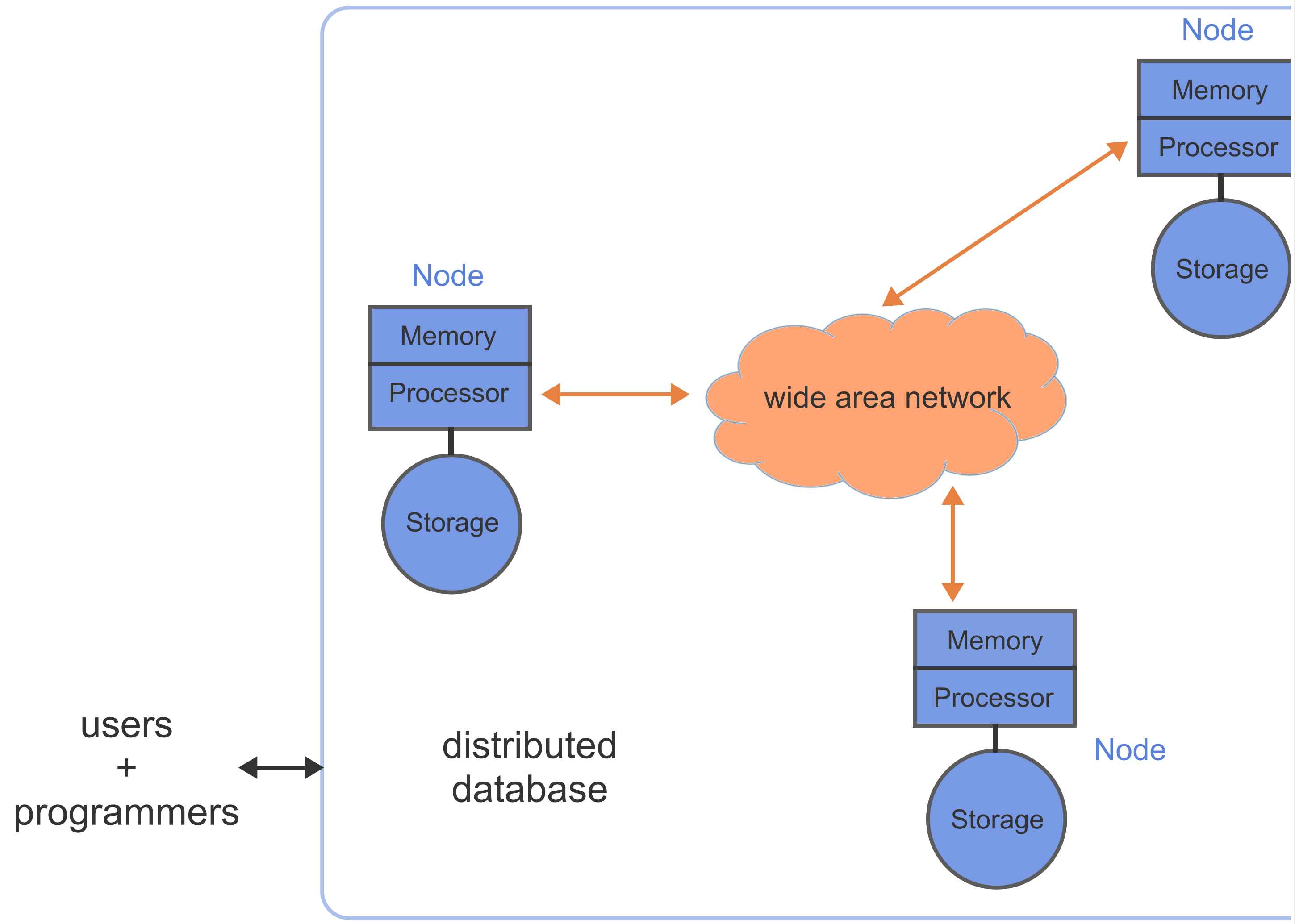
- A ***parallel database*** runs on a parallel computer or cluster.
- A ***distributed database*** runs on multiple computers connected by a wide area network.

Both parallel and distributed databases present a unified view of data to database users and programmers. The physical location of data on storage media is visible to database administrators only.

Internally, parallel and distributed databases behave differently. In a parallel database, data location has limited impact on query processing since local area networks are relatively fast and reliable. In a distributed database, data location is significant since wide area networks are relatively slow and unreliable. Wide area networks create technical challenges with distributed transactions, described unreliable. Wide area networks create technical challenges with distributed transactions, described below.

Despite technical challenges, distributed databases offer compelling benefits for databases with users in many locations. Ex: A company has employees in a dozen locations worldwide. Data for each employee is stored on a local node. Queries about local employees are fast. Queries about remote employees are slower, since both query and results must traverse a wide area network. Typically, queries about remote employees are submitted less often, so slower response is acceptable. All queries are easy to write since data location is invisible to users and programmers.

PARTICIPATION ACTIVITY 22.2.3: Distributed database.



Animation content:

Static figure:
A rectangle named distributed database contains three nodes. The nodes surround a cloud icon, labeled wide area network. Double-headed arrows are between each node and the wide area

network icon. Each node has three parts - memory, processor, and storage. A caption users + database is outside the distributed database rectangle, with a double-headed arrow in between.

Step 1: A node is a single processor with associated memory and storage. One node appears.

Step 2: A distributed database consists of nodes connected by a wide area network. Two more nodes, the wide area network icon, and the distributed database caption appear.

Step 3: From the perspective of database users and programmers, individual nodes of a distributed database are not visible. A rectangle around the three nodes appears. Caption users + database appears. A double-headed arrow appears between users + database and the rectangle.

Animation captions:

1. A node is a single processor with associated memory and storage.
2. A distributed database consists of nodes connected by a wide area network.
3. From the perspective of database users and programmers, individual nodes of a distributed database are not visible.

PARTICIPATION ACTIVITY 22.2.4: Parallel databases.

Indicate if each query can potentially run faster in a parallel database than a database running on a single-processor computer.

1) The Passenger table is stored on one disk drive:

```
SELECT *
FROM Passenger;
```

○ Yes

○ No

2) Passenger table rows are stored on 26 disk drives, depending on the first letter of the passenger's last name:

```
SELECT *
FROM Passenger
ORDER BY LastName;
```

○ Yes

○ No

3) The Employee and Department tables are stored on the same disk drive:

```
SELECT EmployeeName,
DepartmentName
FROM Employee, Department
WHERE Employee.DepartmentCode
= Department.DepartmentCode;
```

○ Yes

○ No

PARTICIPATION ACTIVITY 22.2.5: Distributed databases.

1) A SELECT query may have different results, depending on how data is assigned to nodes of a distributed database.

○ True

○ False

2) An UPDATE query may run faster or slower, depending on how data is assigned to nodes of a distributed database.

○ True

○ False

3) Nodes of a distributed database may share memory or storage.

○ True

○ False

4) Database administrators can optimize query performance by assigning tables or parts of tables to specific nodes of a distributed database.

○ True

○ False

Distributed transactions

A ***distributed transaction*** updates data on multiple nodes of a distributed database. In a distributed transaction, either all nodes or no nodes must be successfully updated. Databases commonly implement distributed transactions with a technique called ***two-phase commit***. The two-phase commit has four steps:

1. In phase 1, a central transaction coordinator notifies all participating nodes of the required updates.
2. Participating nodes receive the notification, store the update in a local log, and send a confirmation message to the transaction coordinator. Participating nodes do not yet commit the update to the database.
3. Phase 2 begins when the transaction coordinator receives confirmation from all participating nodes. The transaction coordinator now instructs all nodes to commit.
4. Participating nodes receive the commit message, commit the update to the database, and notify the transaction coordinator of success.

The two-phase commit must account for the following failure scenarios:

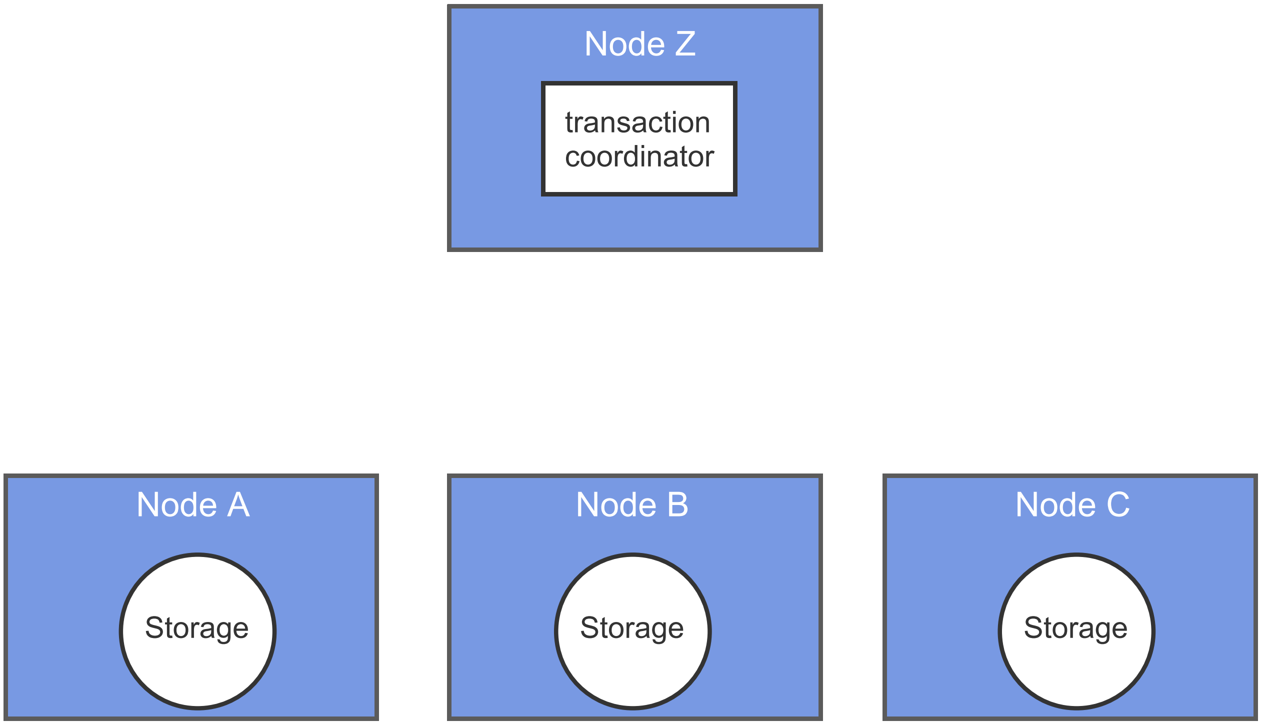
- In step 2, if the transaction coordinator does not receive confirmation from all nodes within a fixed time period, the transaction coordinator instructs participating nodes to roll back the update.
- In step 4, if a node becomes unavailable and fails to notify the transaction coordinator of success, the transaction coordinator resends the commit message until the node responds.

The two-phase commit ensures updates are applied to either all nodes or no nodes. In the first failure scenario, the transaction rolls back, and no updates are applied. In the second failure scenario, the transaction commits, and all updates are applied.

Terminology

Two-phase commit and two-phase locking are different procedures. ***Two-phase commit*** *governs commit and rollback at the end of distributed transactions only.* ***Two-phase locking****, described elsewhere in this material, governs acquisition and release of locks during either local or distributed transactions.*

PARTICIPATION ACTIVITY | 22.2.6: Two-phase commit.



Animation content:

Static figure:
Four nodes appear, with captions node Z, node A, node B, and node C. Node Z contains a transaction coordinator. Nodes A, B, and C contain storage.

Step 1: A transaction coordinator manages updates to distributed nodes. The transaction coordinator may run on any node. All four nodes appear.

Step 2: In phase 1, the transaction coordinator notifies all nodes of updates. Caption phase 1 appears. Actions update A, update B, and update C move from node Z to node A, node B, and node C.

Step 3: Nodes receive updates and send confirmation to the transaction coordinator. Confirmation A, confirmation B, and confirmation C move from node A, node B, and node C to node Z.

Step 4: In phase 2, the transaction coordinator receives all confirmations and instructs nodes to commit. Caption phase 1 is replaced by caption phase 2. Three commit actions move from node Z to nodes A, B, and C.

Step 5: Nodes receive commit messages, save updates to storage, and notify the transaction coordinator of success. The update actions at nodes A, B, and C move into the storage icon within the nodes. Nodes A, B, and C send a success message to node Z.

Animation captions:

1. A transaction coordinator manages updates to distributed nodes. The transaction coordinator may run on any node.
2. In phase 1, the transaction coordinator notifies all nodes of updates.
3. Nodes receive updates and send confirmation to the transaction coordinator.
4. In phase 2, the transaction coordinator receives all confirmations and instructs nodes to commit.
5. Nodes receive commit messages, save updates to storage, and notify the transaction coordinator of success.

PARTICIPATION ACTIVITY 22.2.7: Distributed transactions.

Checking and savings account transactions are stored on different nodes of a distributed database. A database user attempts to deduct $100 from checking and credit $100 to savings. Both debit and credit are processed in a distributed transaction with a two-phase commit.

What is the result of the following scenarios?

1) The node containing the checking transactions fails in phase 1.

○ Both debit and credit are successfully processed.

○ Neither debit nor credit are processed.

○ Both debit and credit are successfully processed, or neither is processed.

○ The result is unpredictable.

2) The node containing the checking transactions fails in phase 2.

○ Both debit and credit are eventually processed.

Both debit and credit are never

○ Both debit and credit are never processed.

○ Debit and credit are either both eventually processed, or both never processed.

○ The result is unpredictable.

Local transactions

A ***local transaction*** updates data on a single node of a distributed database.

Distributed transactions are relatively slow, as multiple nodes must respond before the transaction commits. As a faster alternative, multiple nodes can be updated independently with local transactions:

1. The transaction coordinator notifies participating nodes of required updates.
2. Nodes commit immediately and confirm with the transaction coordinator.
3. If a node is unavailable, the transaction coordinator repeats the update message until confirmation is received.

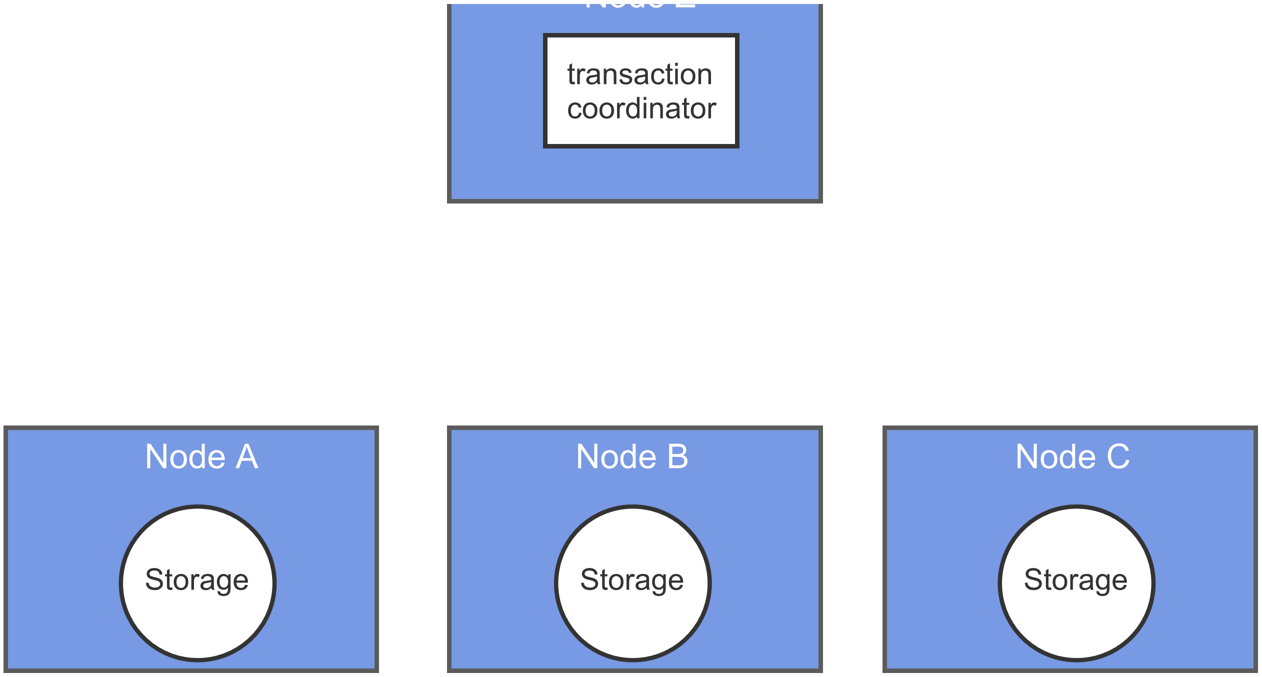
Local transactions create temporary inconsistency, as nodes are updated at different times. The choice of local or distributed transactions depends on performance and consistency requirements. Ex:

- In many financial databases, all nodes must be consistent at all times. Distributed transactions are necessary.
- In databases that log website activity, temporary inconsistency may be acceptable. Local transactions might be used to process updates quickly and support a high volume of web clicks.

Updates in a distributed transaction are ***synchronous***, since the updates occur at the same time from the perspective of the database user. Updates in separate local transactions are ***asynchronous***.

Databases that use local rather than distributed transactions are called ***eventually consistent***.

PARTICIPATION ACTIVITY | 22.2.8: Local transactions.



Animation content:

Static figure:
Four nodes appear, with captions node Z, node A, node B, and node C. Node Z contains a transaction coordinator. Nodes A, B, and C contain storage.

Step 1: Node B fails prior to transactions. Nodes A, B, and C appear. A large X appears above node B.

Step 2: A transaction coordinator notifies all nodes of updates. Node Z appears. Actions update A, update B, and update C move from node Z to node A, node B, and node C.

Step 3: Nodes A and C commit updates and confirm transactions. The update actions at nodes A and C move into the storage icon within the nodes. Nodes A and C send a success message to node Z.

Step 4: Transaction coordinator does not receive confirmation from node B and resends update. Node Z repeatedly sends update B to node B.

Step 5: Node B eventually becomes available, commits update, and confirms transaction. The red X over node B disappears. Node Z sends another update B action to node B, which moves to the node B storage icon. Node B sends a success message to node Z.

Animation captions:

1. Node B fails prior to transactions

1. Node B fails prior to transactions.
2. A transaction coordinator notifies all nodes of updates.
3. Nodes A and C commit updates and confirm transactions.
4. Transaction coordinator does not receive confirmation from node B and resends update.
5. Node B eventually becomes available, commits update, and confirms transaction.

PARTICIPATION ACTIVITY 22.2.9: Local transactions.

Checking and savings account transactions are stored on different nodes of a distributed database. A database user attempts to deduct $100 from checking and credit $100 to savings. The transaction coordinator processes the debit and credit as independent local transactions.

1) The credit and debit updates are:

○ Synchronous

○ Asynchronous

2) The database is:

○ Always consistent

○ Eventually consistent

○ Never consistent

3) The debit node is available. The credit node fails prior to the transactions and becomes available ten minutes later. What is the result?

○ Both debit and credit are successfully processed.

○ Neither debit nor credit are processed.

○ Both debit and credit are successfully processed, or neither is processed.

○ The result is unpredictable.

CAP theorem

A ***consistent*** database conforms to all rules at all times. In a distributed database, a rule may govern data on multiple nodes. Ex: Foreign key values on one node must match primary key values on another node. Ex: Copies of data on multiple nodes must be identical.

In an ***available*** database, 'live' nodes must respond to queries at all times. A 'dead' node may be unresponsive, but 'live' nodes must respond regardless of the state of other nodes.

A ***network partition*** forms when a network error prevents nodes from communicating. A distributed database occasionally experiences network partitions since nodes are connected by wide area networks that occasionally fail. A ***partition-tolerant*** database continues to function when a network partition occurs.

The ***CAP theorem*** states that a distributed database cannot simultaneously be Consistent, Available, and Partition-tolerant. A distributed database can guarantee any two, but not all three, of these properties.

As a practical matter, most distributed databases must always function and are therefore partition-tolerant. Consequently, most distributed databases guarantee either consistency or availability, but not both.

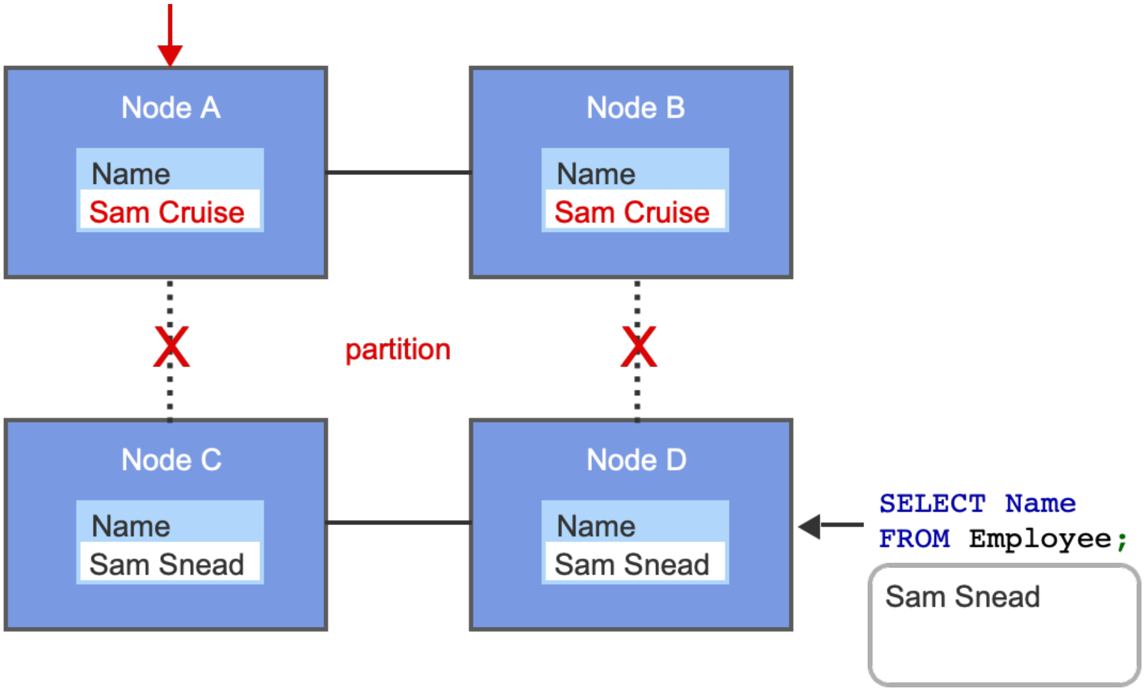
The tradeoff between consistency and availability is relative, not absolute. Since wide area networks are relatively slow, the time to propagate an update from one node to another is significant. If a query accesses updated data before all nodes are updated, the database must either return inconsistent data or not respond immediately. Rather than choose between consistency and availability, a database must choose how long to wait to provide a consistent response.

Terminology

> ***Availability** commonly means the percentage of time a database is responsive to users and programs. In the context of the CAP theorem, however, availability is the response of individual nodes rather than the entire database system.*
>
> *In the context of networks, a **partition** is a subset of nodes. In the context of data storage, a partition is a subset of table data.*

PARTICIPATION ACTIVITY | 22.2.10: CAP theorem.

```
UPDATE Employee
SET Name = 'Sam Cruise';
```



```
SELECT Name
FROM Employee;
```

```
Sam Snead
```

Animation content:

Static figure:
Four nodes appear, labeled A, B, C, and D. Nodes A and B contain a table with column Name and one value, Sam Cruise. Nodes C and D contain a table with column Name and one value, Sam Senad.

The nodes have caption partition. Solid lines connect node A to B and node C to D. Dotted lines covered with an X connect node A to C and node B to D.

An SQL statement appears with an arrow pointing to node A.
Begin SQL code:
UPDATE Employee
SET Name = 'Sam Cruse';
End SQL code.

An SQL statement appears with an arrow pointing to node D.
Begin SQL code:
SELECT Name
FROM Employee;
End SQL code.

A console appears below the SELECT statement and displays Sam Snead.

A console appears below the SELECT statement and displays Sam Snead.

Step 1: A distributed database has four nodes. Each node contains a copy of the Employee table. The four nodes appear with solid lines connecting nodes and no caption. The tables in all four nodes contain the value Sam Snead.

Step 2: The wide area network fails, creating a partition. Nodes A and B cannot communicate with nodes C and D. The caption partition appears. The lines connecting node A to C and B to C become dotted. An X appears above these lines.

Step 3: Nodes A and B are updated with local transactions. The partition prevents updates to nodes C and D. The UPDATE statement appears above node A. The values in nodes A and B change to Sam Cruise.

Step 4: Node D receives an Employee query. The SELECT statement appears next to node D.

Step 5: If node responds, the result is inconsistent. If node does not respond, the database is unavailable. The value Sam Snead appears in the console.

Animation captions:

1. A distributed database has four nodes. Each node contains a copy of the Employee table.
2. The wide area network fails, creating a partition. Nodes A and B cannot communicate with nodes C and D.
3. Nodes A and B are updated with local transactions. The partition prevents updates to nodes C and D.
4. Node D receives an Employee query.
5. If node responds, the result is inconsistent. If node does not respond, the database is unavailable.

PARTICIPATION ACTIVITY | 22.2.11: Consistency, availability, and partition-tolerance.

Match the property to the description.

If unable to drag and drop, refresh the page.

Not partition-tolerant | Not consistent | Not available

Every millisecond, a master node

sends a message to all other nodes. The master node waits a millisecond for replies but does not hear from node X. The master node reroutes the message through other nodes and waits another millisecond. The master node still does not hear from node X and suspends processing on all nodes.

Node B contains a copy of node A data. Node B receives a query and requests the latest version of node A data, but node A does not respond. Node B is not certain the copy is current and does not execute the query.

Node B contains a copy of node A data. Node A is updated in a local transaction. After the transaction commits, the database instructs node B to update the copy in a separate local transaction.

Reset

CHALLENGE ACTIVITY | 22.2.1: Distributed databases.

544874.3500394.qx3zqy7

Start

Indicate whether each configuration is shared memory, shared storage, or shared nothing.

Pick (a) A file system allows direct disk access from multiple computers
time can access the file system and the shared disk can be connec
channel to ensure adequate I/O performance.

Pick (b) A graphics processing unit (GPU) runs hundreds to thousands c organized in blocks.

Pick

(c) A bank stores data into a database management system where local disks to minimize the interference of resource sharing. Proces communicate with another processor using an interconnection net

1	2	3	4

Check Next

Exploring further:

- Parallel computing
- Two-phase commit
- CAP theorem

22.3 Replicated databases

Replicas

A ***replica*** is a copy of an entire database, a table, or a subset of table data. A ***replicated database*** maintains two or more replicas on separate storage devices. Data can be replicated in any database with multiple storage devices, such as parallel and distributed databases.

Replicated databases have several major advantages:

- *High availability*. If one storage device fails, the database routes queries to a replica on another storage device. In general, if a database maintains N replicas, the database can survive simultaneous failure of N-1 storage devices.
- *Fast concurrent reads*. Concurrent queries can read separate replicas without interfering with each other. One large query can be decomposed into smaller queries that read separate replicas in parallel.
- *Local reads*. In a distributed database, reads can be executed locally, eliminating network

- *Local reads*. In a distributed database, reads can be executed locally, eliminating network delays and outages.

Replicated databases have one major disadvantage:

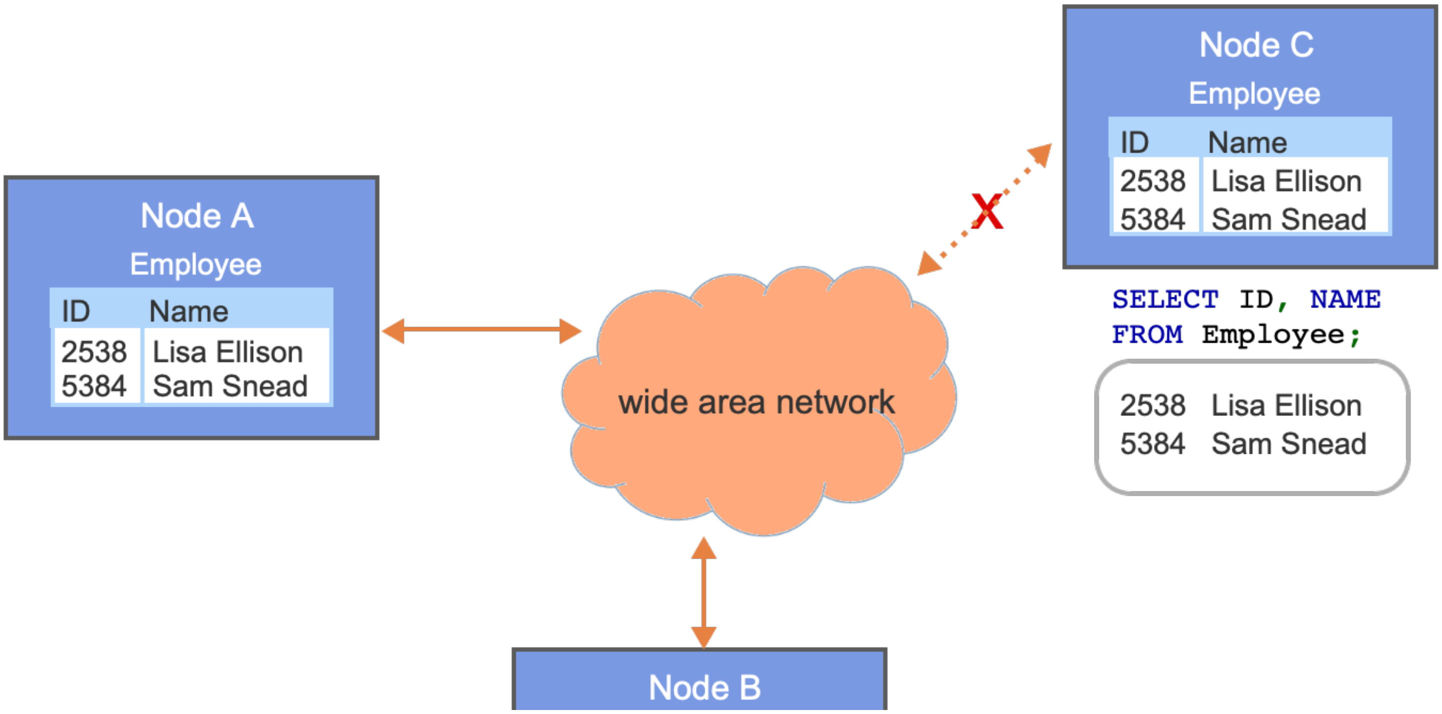
- *Slow or inconsistent updates*. Updates must be applied to all replicas on multiple storage devices. If all replicas on different nodes are updated with a distributed transaction, the update is relatively slow. If replicas on different nodes are updated with local transactions, updates are relatively fast but replicas are temporarily inconsistent.

Replication simplifies some database administration activities but makes others more complex:

- *Simple backup*. One replica can be backed up while transactions execute against other replicas.
- *Enhanced security*. Updates can be restricted to one replica, accessible only to trusted database users. Updates are propagated to read-only replicas, accessible to a broader user group.
- *Complex server administration*. Database administrators must determine how to propagate updates across replicas.

Replication is commonly used in parallel and distributed databases, particularly when reads are frequent, updates are infrequent, and temporary inconsistency is acceptable.

PARTICIPATION ACTIVITY 22.3.1: Replicated database advantages.

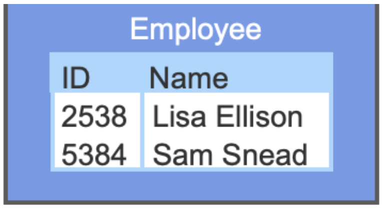


ID	Name
2538	Lisa Ellison
5384	Sam Snead

ID	Name
2538	Lisa Ellison
5384	Sam Snead

```
SELECT ID, NAME
FROM Employee;
```

```
2538   Lisa Ellison
5384   Sam Snead
```



Employee

ID	Name
2538	Lisa Ellison
5384	Sam Snead

Animation content:

Static figure:
Three nodes are named node A, node B, and node C. The nodes surround a cloud icon, labeled wide area network. Double-headed arrows are between each node and the cloud icon. The arrow between the cloud and Node C is dotted and covered by an X. The other arrows are solid with no X. Each node contains an identical Employee table.

An SQL statement appears below Node C.
Begin SQL code:
SELECT ID, Name
FROM Employee;
End SQL code.

The Employee rows appear on a console below the SQL statement.

Step 1: The Employee table is stored on node A of a distributed database and not replicated. All nodes and the wide area network icon appear. All double-headed arrows are solid. The Employee table appears inside node A only.

Step 2: An Employee query at node C generates a round-trip over the wide area network. The query is relatively slow. The SELECT statement appears below node C and moves through the wide area network to node A. The table in node A moves through the wide-area network and appears on the console below node C.

Step 3: If a network partition occurs, the query cannot be processed. The arrow between node C and the wide area network becomes dotted with an X. The SELECT statement appears again below node C but cannot move through the wide area network to node A. The console below node C is blank.

Step 4: If the Employee table is replicated, the query is local, relatively fast, and immune to a network partition. Copies of the Employee table appear in nodes B and C. The SELECT statement appears below node C. The table moves from node C and appears on the console.

Animation captions:

1. The Employee table is stored on node A of a distributed database and not replicated.
2. An Employee query at node C generates a round-trip over the wide area network. The query is relatively slow.
3. If a network partition occurs, the query cannot be processed.
4. If the Employee table is replicated, the query is local, relatively fast, and immune to a network partition.

PARTICIPATION ACTIVITY | 22.3.2: Replication advantages.

'Local reads' is an advantage of replicated databases in some, but not all, configurations. Which configurations experience faster local reads when data is replicated?

1) A database running on a single node with multiple storage devices.

○ True

○ False

2) A parallel database running on a shared memory parallel computer.

○ True

○ False

3) A parallel database running on a shared nothing computer cluster.

○ True

○ False

4) A distributed database.

○ True

○ False

Updating replicas

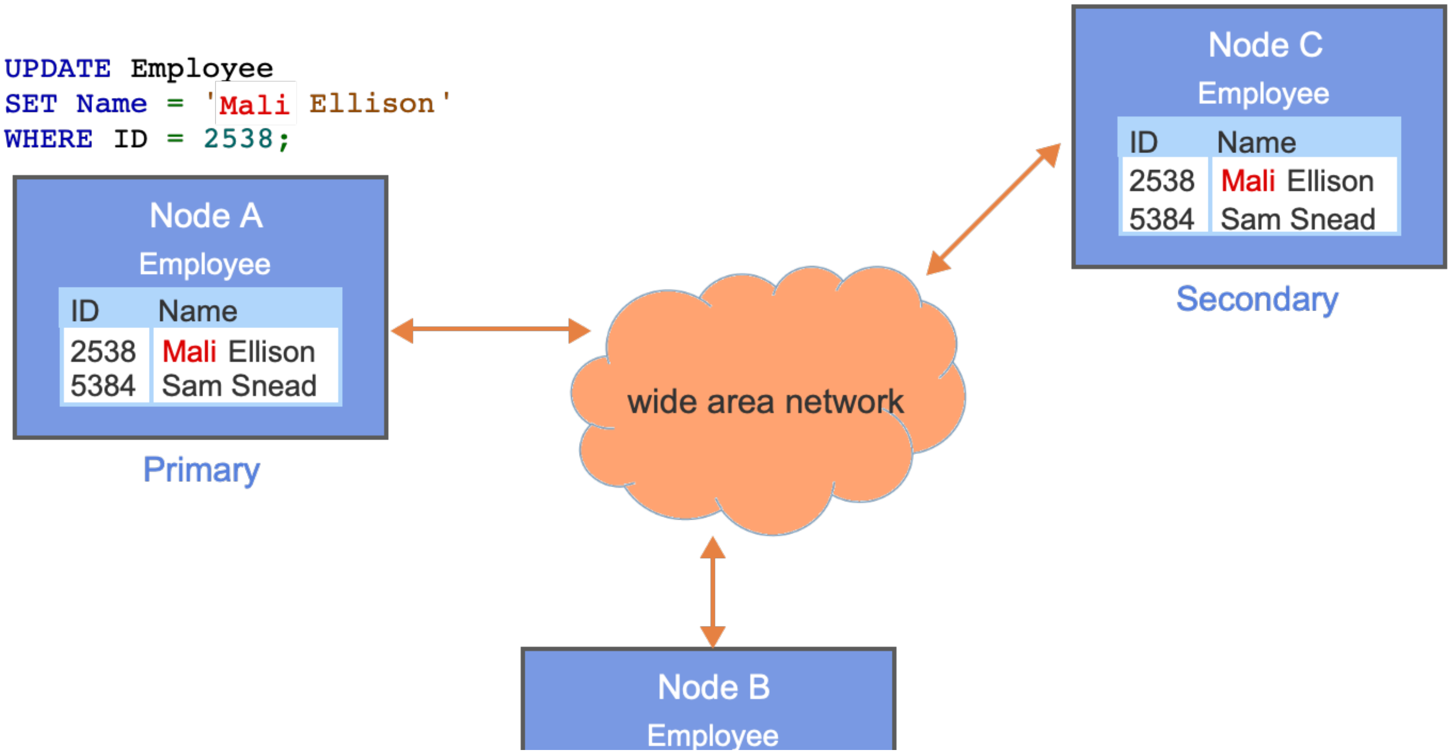
Updating replicated data in a database running on a single node is straightforward. Some storage devices, called ***storage arrays***, manage replicas internally, without database intervention. Alternatively, the database can update all replicas within a single local transaction. Either way, synchronizing replicas does not require special database capabilities.

Updating replicated data in a distributed database is more complex. Updating all replicas in a distributed transaction guarantees consistency but is relatively slow and fails when any replica is unavailable. Two alternative techniques are commonly used:

- The ***primary/secondary*** technique designates one node as primary. All updates are first applied to the primary node in local transactions. Secondary nodes are updated after the primary node commits, with independent local transactions. If the primary node fails, the database automatically designates a new primary node to ensure continued availability.
- The ***group replication*** technique applies updates to any node in a group. Prior to committing, a node broadcasts transaction information to other nodes, which look for conflicts with concurrent transactions. If any node detects a conflict, an algorithm determines which transaction commits and which rolls back. This algorithm may be simple, such as the transaction that commits first wins, or complex. If a network partition occurs and nodes cannot communicate, processing is temporarily suspended.

Support for these techniques varies across relational databases. MySQL with the InnoDB storage engine supports both techniques as well as distributed transactions.

PARTICIPATION ACTIVITY 22.3.3: Primary/secondary replication.



```
UPDATE Employee
SET Name = 'Mali Ellison'
WHERE ID = 2538;
```

Node A

Employee

ID	Name
2538	Mali Ellison
5384	Sam Snead

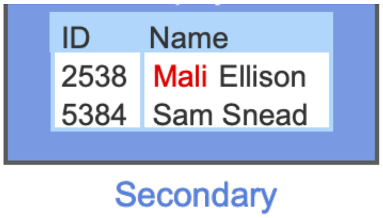
Primary

Node C

Employee

ID	Name
2538	Mali Ellison
5384	Sam Snead

Secondary



ID	Name
2538	Mali Ellison
5384	Sam Snead

Animation content:

Static figure:
Three nodes are named node A, node B, and node C. Node A is labeled primary. Nodes B and C are labeled secondary. The nodes surround a cloud icon, labeled wide area network. Double-headed arrows are between each node and the cloud icon. Each node contains an identical Employee table. The first row of the table contains the name Mali Ellison.

An SQL statement appears above node A.
Begin SQL code:
UPDATE Employee
SET Name = 'Mali Ellison'
WHERE ID = 2538;
End SQL code.
In this statement, the given name Mali is highlighted.

Step 1: The Employee table is replicated on three nodes. All three nodes appear containing identical Employee tables. The first row of the tables contain the name Lisa Ellison. The cloud icon appears.

Step 2: Node A is primary. The label primary appears under node A. The label secondary appears under nodes B and C.

Step 3: Updates are applied to the primary node only, so the database is temporarily inconsistent. The UPDATE statement appears above node A. Lisa changes to Mali in the table in node A.

Step 4: After update commits on primary node, secondary nodes are updated in local transactions. Mali moves through the wide area network to nodes B and C. Lisa changes to Mali in the tables in nodes B and C.

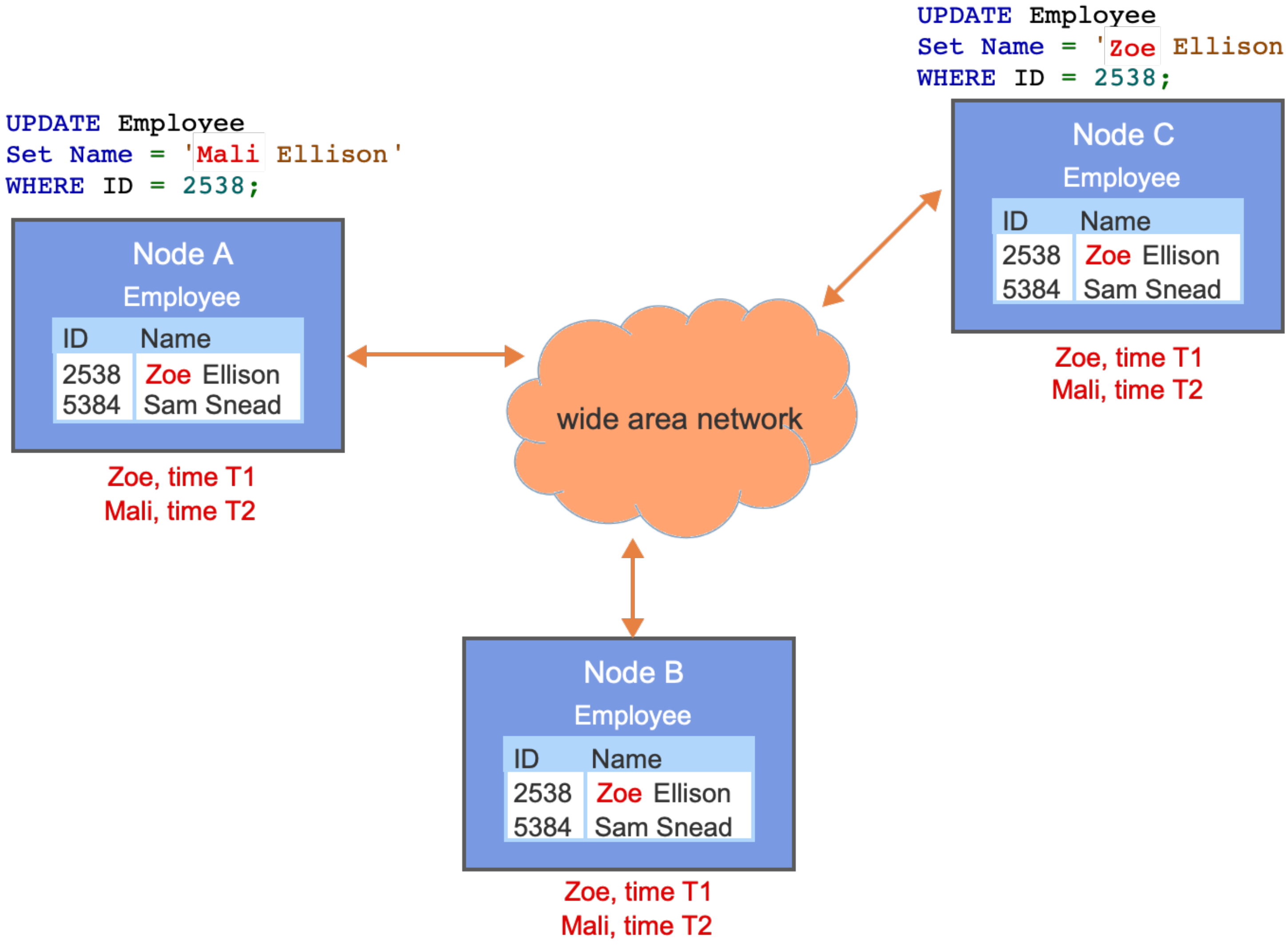
Animation captions:

1. The Employee table is replicated on three nodes.

1. The Employee table is replicated on three nodes.
2. Node A is primary.
3. Updates are applied to the primary node only, so the database is temporarily inconsistent.
4. After update commits on primary node, secondary nodes are updated in local transactions.

PARTICIPATION ACTIVITY

22.3.4: Group replication.



```
UPDATE Employee
Set Name = 'Mali Ellison'
WHERE ID = 2538;
```

```
UPDATE Employee
Set Name = 'Zoe Ellison
WHERE ID = 2538;
```

ID	Name
2538	Zoe Ellison
5384	Sam Snead

Animation content:

Static figure:
Three nodes are named node A, node B, and node C. The nodes surround a cloud icon, labeled wide area network. Double-headed arrows are between each node and the cloud icon. Each node contains an identical Employee table. Below all nodes is caption Zoe - time T1, followed by the caption Mali - time T2.

caption Mali - time T2.

An SQL statement appears above node A.
Begin SQL code:
UPDATE Employee
SET Name = 'Mali Ellison'
WHERE ID = 2538;
End SQL code.
In this statement, the given name Mali is highlighted.

Another SQL statement appears above node C.
Begin SQL code:
UPDATE Employee
SET Name = 'Zoe Ellison'
WHERE ID = 2538;
End SQL code.
In this statement, the given name Zoe is highlighted.

Step 1: The Employee table is replicated on three nodes. All nodes and the wide area network appear. All nodes contain the same Employee table, with the name Lisa Ellison.

Step 2: Any node in the group can be updated. The UPDATE statements appear above nodes A and C.

Step 3: Prior to commit, the updated node sends transaction information to all other nodes. Caption 'Zoe, time T1' appears below node C and moves through the wide area network to nodes A and B. Caption 'Mali, time T2' appears below node A and moves through the wide area network to nodes B and C.

Step 4: All nodes detect a conflict. Since Zoe update committed prior to Mali update, Zoe wins. In all nodes, Lisa Ellison changes to Zoe Ellison.

Animation captions:

1. The Employee table is replicated on three nodes.
2. Any node in the group can be updated.
3. Prior to commit, the updated node sends transaction information to all other nodes.
4. All nodes detect a conflict. Since Zoe update committed prior to Mali update, Zoe wins.

PARTICIPATION ACTIVITY 22.3.5: Updating replicas.

Match the update technique with the description. For definitions of 'eventually consistent' and 'partition-tolerant', see the material on distributed databases.

If unable to drag and drop, refresh the page.

Group replication **Distributed transaction** **Primary/secondary**

	Eventually consistent, not partition-tolerant.
	Always consistent, not partition-tolerant.
	Eventually consistent, partition-tolerant.

Reset

CHALLENGE ACTIVITY 22.3.1: Replicated databases.

544874.3500394.qx3zqy7

Start

Which of the following configurations experience faster local reads when data is replicatec

□ (a) A small sized financial organization hosts data and applications in one server roon

□ (b) An international online library with many branches keeps a record of all books own

□ (c) Intensive care unit (ICU) health data is stored within the hospital.

□ (d) A local shop stores and manages in-house supply and order data.

□ (e) A data center consists of multiple computers in a centralized location.

1 2 3

Check Next

Replicated catalogs

A catalog is a directory of information describing database objects such as tables, columns, keys, and indexes. Catalog information is necessary to process queries and access data. Each node in a distributed database can process queries and therefore requires access to the catalog.

In a distributed database, the catalog can be structured in two ways:

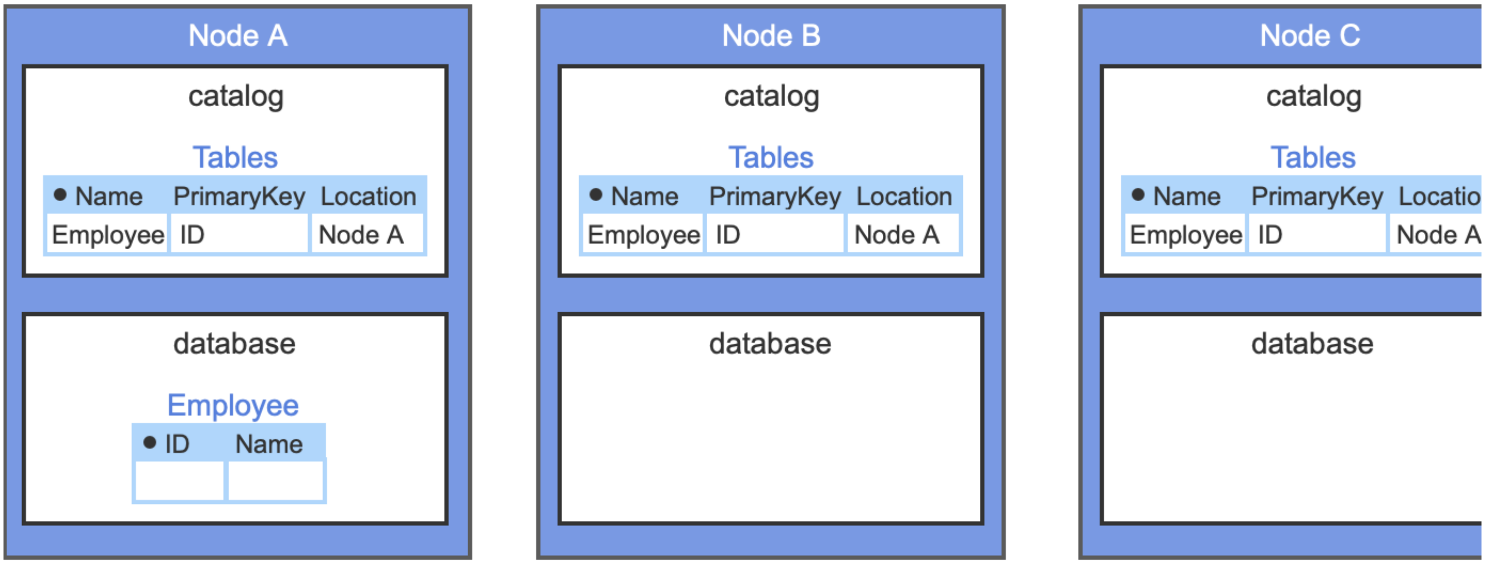
- In a ***central catalog***, the entire catalog resides on a single node. Storing the catalog on a single node is relatively easy to manage. However, query processing at remote nodes must access the catalog via a wide area network, which may be slow or unreliable. Furthermore, query processing at all nodes interact with the central catalog, which may become a bottleneck.
- In a ***replicated catalog***, a copy of the catalog resides on each node. Most queries are fast and reliable since all catalog data is available locally. However, statements that update the catalog, such as CREATE, ALTER, and DROP, must update all replicas. Updating replicas generates increased network traffic and, if executed in a distributed transaction, fails when any replica is unavailable.

Since catalog updates are infrequent compared to other database queries, many distributed databases use a replicated catalog. To improve performance of catalog updates, many databases use a variation of the primary/secondary technique. Updates are first applied to the replica on the node containing the affected data object and then propagated to other replicas.

When catalog replicas are updated with local transactions, some replicas are momentarily out of date. If a query cannot be processed due to an out-of-date replica, the database might display an error and advise the user to resubmit the query. This rarely occurs, however, since catalog updates are infrequent and the delay between replica updates is short.

PARTICIPATION ACTIVITY

22.3.6: Replicated catalog.



Tables

● Name	PrimaryKey	Location
Employee	ID	Node A

Tables

● Name	PrimaryKey	Location
Employee	ID	Node A

Tables

● Name	PrimaryKey	Locatio
Employee	ID	Node A

Employee

● ID	Name

```
CREATE TABLE Employee (
  ID INT,
  Name VARCHAR(20),
  PRIMARY KEY (ID)
);
```

Animation content:

Static figure:
Nodes A, B and C appear. All three nodes contain an icon labeled catalog and another icon labeled database. In the catalog in all three nodes, a table named Tables appears, with columns Name, PrimaryKey, and Location. All tables have the same row:
Employee, ID, Node A

In node A, the database contains the table Employee with primary key ID. In nodes B and C, the database is empty.

An SQL statement appears below node A.
Begin SQL code:

```
CREATE TABLE Employee (
 ID INT,
 Name VARCHAR(20),
 PRIMARY KEY (ID)
);
```

);
End SQL code.

Step 1: A replicated catalog is stored in each node of a distributed database. Nodes A, B, and C appear. The catalogs and databases are empty.

Step 2: The catalog contains a table called 'Tables', with one row describing each table in the database. In the catalog in all nodes, the Tables table appears with no rows.

Step 3: The administrator creates a table on node A. The syntax for assigning tables to nodes varies and is not shown. The CREATE TABLE statement appears below node A.

Step 4: A transaction on node A creates the table and inserts a row into Tables. In the node A catalog, the Employee row appears in the Tables table. In the node A database, the Employee table appears.

Step 5: After node A transaction commits, separate transactions update node B and C catalogs. The Employee row is duplicated on node A and moves to Tables in the node B and C catalogs. The databases on nodes B and C remain empty.

Animation captions:

1. A replicated catalog is stored in each node of a distributed database.
2. The catalog contains a table called 'Tables', with one row describing each table in the database.
3. The administrator creates a table on node A. The syntax for assigning tables to nodes varies and is not shown.
4. A transaction on node A creates the table and inserts a row into Tables.
5. After node A transaction commits, separate transactions update node B and C catalogs.

PARTICIPATION ACTIVITY | 22.3.7: Distributed catalogs.

Refer to the above animation. A database administrator submits a CREATE TABLE statement at node C, assigning the table to node A.

1) What is the result if node A is unavailable?

○ The CREATE TABLE statement succeeds.

○ The CREATE TABLE statement

○ fails.

○ The catalogs on node B and C are updated. The catalog on node A is updated when node A becomes available.

2) What is the result if node B is unavailable?

○ The CREATE TABLE statement succeeds.

○ The CREATE TABLE statement fails.

○ The result is unpredictable.

3) Suppose the catalog is central, not replicated, and stored on node A. Node A is unavailable. A user submits a SELECT statement on node C that references only C tables.

○ The SELECT statement succeeds.

○ The SELECT statement fails.

○ The result is unpredictable.

Exploring further:

- MySQL primary/secondary replication
- MySQL group replication
- MySQL replication statements

22.4 Data warehouses

Operational and analytic data

Organizations use ***operational data*** to conduct daily business functions. Ex: Sales invoices, student

Organizations use ***operational data*** to conduct daily business functions. Ex: Sales invoices, student test scores, and driving violation records are operational data. Organizations use ***analytic data*** to understand, manage, and plan the business. Ex: Sales totals by region, average student grades over time, and driving violation counts by ZIP code are analytic data. Analytic data is sometimes called ***reporting data*** or ***decision support data***.

Operational and analytic data differ in several ways:

- *Volatility*. Operational data changes in real time as business functions are executed. Analytic data is updated at fixed intervals, often daily or weekly, so that reports and summaries always refer to a known time.
- *Detail*. Most operational data is detailed, reflecting individual transactions. Analytic data is often summarized by time period, business unit, geography, and other business dimensions.
- *Scope*. Most operational databases are designed for a specific business function. Consequently, operational databases supporting different business functions are often incompatible. Analytic databases combine data from many business functions in an integrated, enterprise-wide view of data, with standard formats, data types, and keys across all tables.
- *History*. Many operational databases are concerned primarily with current data. Analytic databases often track trends over time and therefore usually contain current and historic data. Ex: Operational data may include active employees only. Analytic data may include past employees and illustrate changes in total employment by month.

Because of the above differences, operational and analytic data are often maintained in separate databases with different designs.

Figure 22.4.1: Operational and analytic data characteristics.

	Volatile	Detailed	Summary	Enterprise-wide	Historical
Operational data	✔	✔			
Analytic data		✔	✔	✔	✔

PARTICIPATION ACTIVITY 22.4.1: Operational and analytic data.

The examples below describe data in several operational databases for a company. Match each data characteristic with the example that illustrates the characteristic.

Match each data characteristic with the example that illustrates the characteristic.

If unable to drag and drop, refresh the page.

Detailed Volatile Historic Summary Enterprise-wide

	A database stores each line item of an order, including item name, product code, cost, and quantity.
	The database records the current project assignments for each employee. Updates are made right away when assignments change. The database does not retain prior assignments.
	Each year, employees receive raises for performance and increased cost of living. The database retains salary data for every year employees have worked at the company.
	The database stores total count of employees working at each corporate facility.
	A report lists employee names alongside employee office numbers. Employee names are extracted from a human resources database. Employee office assignments are stored in a separate facilities database.

Reset

Data warehouses

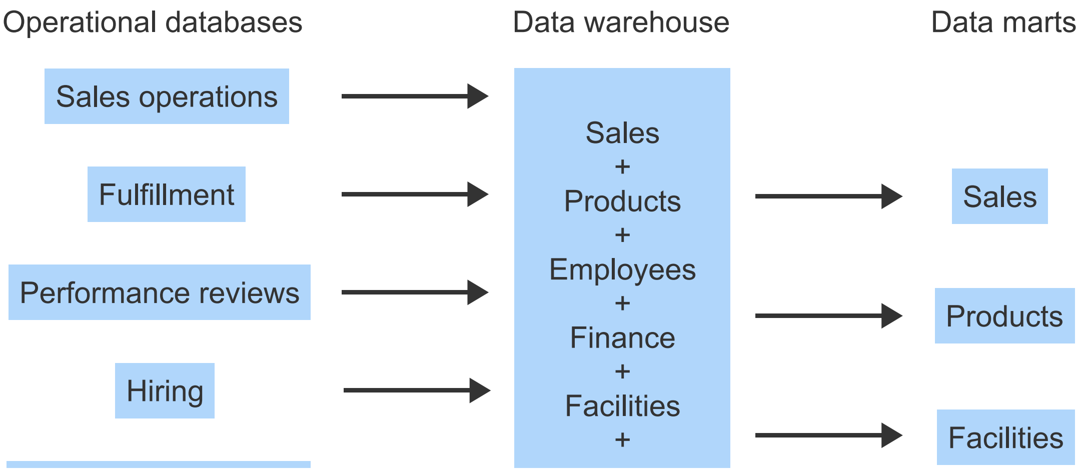
Storing operational and analytic data in the same database creates several problems:

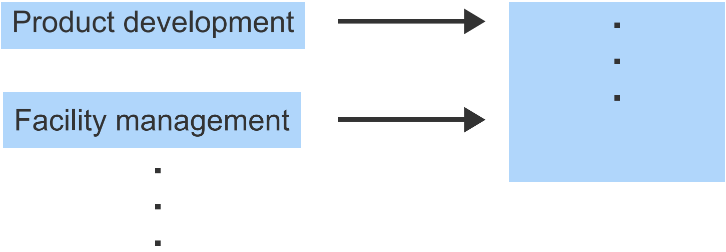
- *Database design*. Since operational data is volatile, operational databases are typically optimized for updates, with most tables in third normal form. Third normal form minimizes redundancy but generates many tables and is not optimal for analytic queries. Analytic queries often combine columns from many third normal form tables, resulting in complex joins that are difficult to write and slow to run.
- *Interference*. Analytic queries often summarize large volumes of data. When executed against an operational database, analytic queries compete with operational queries, degrade query response time, and interfere with business operations.
- *Reference time*. Analytic queries usually reference a specific point in time, such as sales totals as of midnight on the last day of the month. Since operational data is volatile, results depend on the precise time a query is submitted. Analytic queries against operational databases thus have an uncertain reference time and may be misleading.

A ***data warehouse*** is a separate database optimized for analytics rather than operations. A data warehouse consists of data extracted from operational databases and restructured to support analytic queries. Data is usually extracted periodically, at a fixed time, so that data in the warehouse has a known reference time. Data is extracted during times of low database use to minimize impact on operational queries.

Data warehouses integrate data from multiple business functions for use by the entire organization. A ***data mart*** is a data warehouse designed for a specific business area, such as sales, human resources, or product development. Since data marts have smaller scope than a data warehouse, data marts are easier to build and maintain. A data mart can be derived directly from operational databases or indirectly from a data warehouse.

PARTICIPATION ACTIVITY 22.4.2: Data warehouse and data marts.





Animation content:

Static figure:
Three diagrams appear with captions operational databases, data warehouse, and data marts.

The operational databases diagram contains six database icons, with captions sales operations, fulfillment, performance reviews, hiring, product development, and facility management. The data warehouse diagram contains one database with caption "sales + products + employees + finance + facilities + . . ." The data marts diagram contains three databases with captions sales, products, and facilities.

An arrow points from each operational database to the data warehouse. An arrow points from the data warehouse to each data mart.

Animation captions:

1. Different business functions use separate operational databases with incompatible keys and data formats.
2. Periodically, data is extracted from operational databases, restructured, and loaded into a data warehouse.
3. Data in the warehouse is integrated, with compatible keys and a standard format.
4. Data marts support specific areas of data and can be derived from the data warehouse.

PARTICIPATION ACTIVITY 22.4.3: Data warehouses.

An organization runs analytic queries against an operational database. Database users report the problems listed below. Select the probable cause of each problem.

1) A telephone company employee, located in the Eastern Time Zone, runs a report that counts telephone calls by hour and area code. The report always runs quickly during

early mornings but slowly during late mornings.

○ Database design

○ Interference

○ Reference time

2) A medical assistant for a large healthcare company reports the percentage of each type of illness over all hospitals. The report always runs slowly, regardless of day or time.

○ Database design

○ Interference

○ Reference time

3) Just before a meeting, two employees generate a report of total sales revenue by day and state. The employees run the same query but the reports conflict.

○ Database design

○ Interference

○ Reference time

Extract, transform, load

Data warehouses are refreshed periodically with a five-step process:

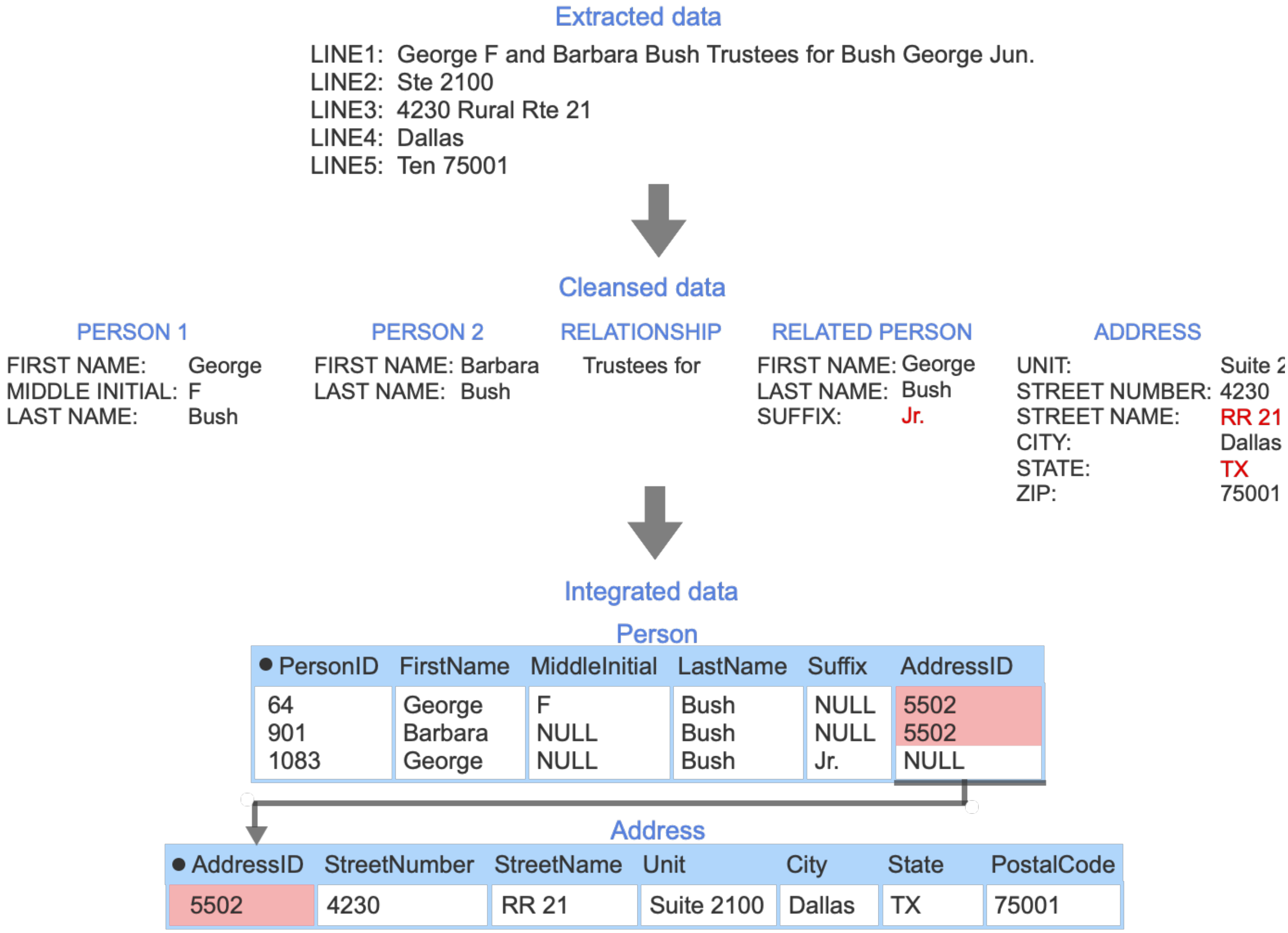
1. *Extract* data from operational databases into a temporary database, called a 'staging area'. Since the data warehouse already contains data from the prior period, only data that has changed since the prior period is extracted.
2. *Cleanse* data to eliminate errors, unusual spellings and incorrect data. Ex: Apply standard abbreviations to addresses, such as RD for road and AVE for avenue. Ex: Abbreviate middle name to the first initial.
3. *Integrate* data into a uniform structure. Ex: Convert all length data to the metric system. Ex: Replace incompatible primary and foreign keys with consistent values.
4. *Restructure* data into a design optimized for analytic queries.

5. *Load* data to the data warehouse.

The five-step process is commonly referred to as the ***extract-transform-load***, or ***ETL***, process. Since the ETL process is time-consuming and difficult to automate, many organizations use special software products, called ***ETL tools***, to minimize programming. Dozens of commercial and open source ETL tools are available, such as:

- *PowerCenter* from Informatica is a high-end ETL product intended to manage large extracts for complex organizations.
- *SQL Server Integration Services* from Microsoft is designed for SQL Server data warehouses.
- *Oracle Data Integrator* supports many data sources but is optimized to load Oracle database products.

PARTICIPATION ACTIVITY 22.4.4: Data cleansing and integration.



Extracted data

LINE1: George F and Barbara Bush Trustees for Bush George Jun.
LINE2: Ste 2100
LINE3: 4230 Rural Rte 21
LINE4: Dallas
LINE5: Ten 75001

Cleansed data

PERSON 1
FIRST NAME: George
MIDDLE INITIAL: F
LAST NAME: Bush

PERSON 2
FIRST NAME: Barbara
LAST NAME: Bush

RELATIONSHIP
Trustees for

RELATED PERSON
FIRST NAME: George
LAST NAME: Bush
SUFFIX: Jr.

ADDRESS
UNIT: Suite 2
STREET NUMBER: 4230
STREET NAME: RR 21
CITY: Dallas
STATE: TX
ZIP: 75001

Integrated data

Person

● PersonID	FirstName	MiddleInitial	LastName	Suffix	AddressID
64	George	F	Bush	NULL	5502
901	Barbara	NULL	Bush	NULL	5502
1083	George	NULL	Bush	Jr.	NULL

Address

● AddressID	StreetNumber	StreetName	Unit	City	State	PostalCode
5502	4230	RR 21	Suite 2100	Dallas	TX	75001

Animation content:

Static figure:
Three diagrams appear, with captions extracted data, cleansed data, and integrated data. Arrows indicate that extracted data is converted to cleansed data, which is converted to integrated data.

The extracted data contains five lines:
LINE1: George F and Barbara Bush Trustees for Bush George Jun.
LINE2: Ste 2100
LINE3: 4230 Rural Rte 21
LINE4: Dallas
LINE5: Ten 75001

The cleansed data contains captions person 1, person 2, relationship, related person, and address. Three lines appear under person 1:
FIRST NAME: George
MIDDLE INITIAL: F
LAST NAME: Bush

Two lines appear under person 2:
FIRST NAME: Barbara
LAST NAME: Bush

One line appears under relationship:
Trustees for

Three lines appear under related person:
FIRST NAME: George
LAST NAME: Bush
SUFFIX: Jr.

Five lines appear under address:
UNIT: Suite 2100
STREET NUMBER: 4230
STREET NAME: RR 21
CITY: Dallas
STATE: TX
ZIP: 75001

The integrated data contains tables Person and Address. Person has columns PersonID, FirstName, MiddleInitial, LastName, Suffix, and AddressID. PersonID is the primary key. Person has three rows:
61, George, F, Bush, NULL, 5502
901, Barbara, NULL, Bush, NULL, 5502
1083, George, NULL, Bush, Jr., NULL

Address has columns AddressID, StreetNumber, StreetName, Unit, City, State, PostalCode. AddressID is the primary key. Address has one row:
5502, 4230, RR 21, Suite 2100, Dallas, TX, 75001

An arrow points from AddressID in the Address table to AddressID in the Person table.

Step 1: Data is extracted from an operational database containing mailing addresses. LINE5 contains an error - the state should be Texas. The extracted data appears.

Step 2: Data is unpacked into separate fields. The cleansed data appears without its caption. The following data has not yet been cleansed:
Under related person, Jun appears instead of Jr.
Under address, Rural Rte 21 appears instead of RR 21.
Under address, Ten appears instead of TX.

Step 3: When cleansing data, standard abbreviations replace source data. Jr replaces Jun, RR 21 replaces R Rte 21, and TN replaces Ten.

Step 4: Postal code 75001 and city Dallas are in Texas, not Tennessee. TX replaces TN. The caption cleansed data appears.

Step 5: Cleansed data is loaded to data warehouse tables. If necessary, new primary and foreign keys values are generated. The integrated data appears. The AddressID columns in both tables are highlighted.

Animation captions:

1. Data is extracted from an operational database containing mailing addresses. LINE5 contains an error - the state should be Texas.
2. Data is unpacked into separate fields.
3. When cleansing data, standard abbreviations replace source data.
4. Postal code 75001 and city Dallas are in Texas, not Tennessee.
5. Cleansed data is loaded to data warehouse tables. If necessary, new primary and foreign keys values are generated.

PARTICIPATION ACTIVITY

22.4.5: Extract, transform, load.

The examples below describe steps in populating a data warehouse for a college. Match the step with the example that illustrates the step.

If unable to drag and drop, refresh the page.

Cleanse Extract Load Integrate Restructure

	Student test scores and attendance records are read from a learning management system and written to a temporary database for further processing.
	Student final grades in the temporary database are compared to data stored in the registrar's database. Discrepancies are reported and reconciled by the registrar's office.
	Grade records in the temporary database include course names but not official course numbers. Course numbers are looked up in an online course catalog and replace course names in student grade records.
	Student addresses are extracted from student grade records and stored in a separate address table.
	Each Saturday, at midnight, new data from the prior week is added to the college data warehouse.

Reset

CHALLENGE ACTIVITY | 22.4.1: Data warehouses.

544874.3500394.qx3zqy7

Start

The following describe data in a company's database. Match each data characteristic witł illustrates the characteristic.

Pick ⇕ (a) Managers schedule annual reviews for every employee in the cc the employees' performance. The database retains performance dạ department to reevaluate employees' compensation.

Pick ⇕ (b) The database stores each employee working in the company, in employee's name, position and salary.

Based on the above characteristics, is the data operational or analytic? Pick ⇕

1	2	3

Check Next

Exploring further:

- Leading ETL tools

22.5 Data warehouse design

22.5 Data warehouse design

Dimensional design

To simplify analytic queries, data warehouses commonly use a dimensional design. A ***dimensional design***, also called a ***star schema***, consists of fact and dimension tables:

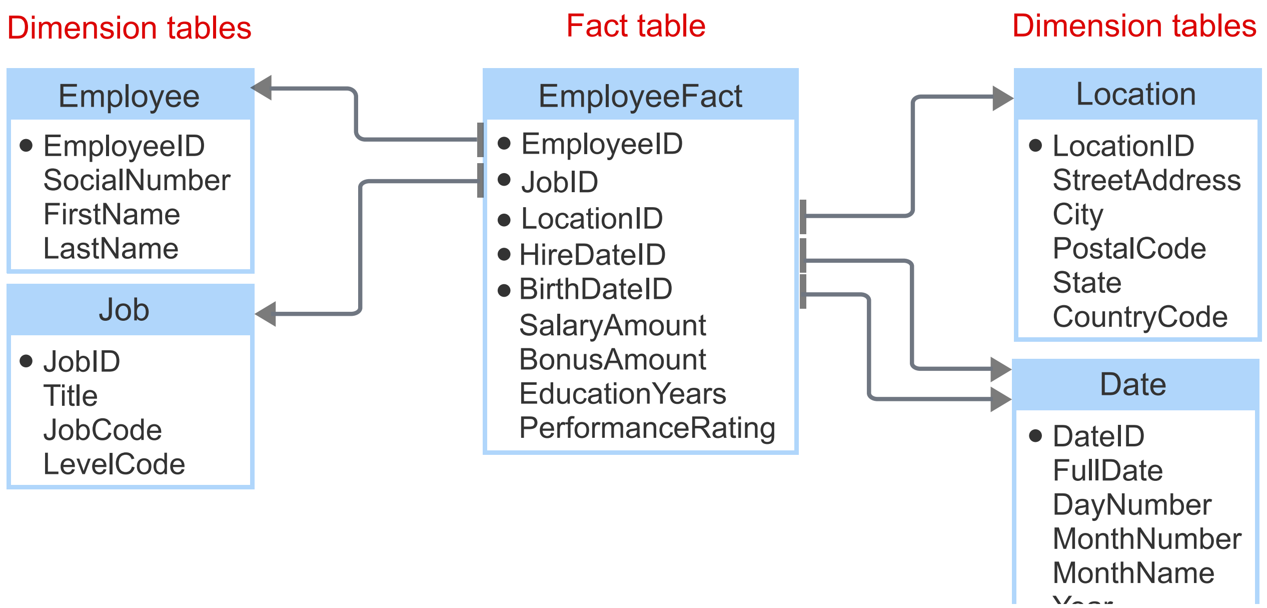
- A ***fact table*** contains numeric data used to measure business performance, such as sales revenue or number of employees. Each row in a fact table consists of numeric fact columns and foreign keys that reference dimension tables.
- A ***dimension table*** contains textual data that describes the fact data, such as product line, organizational unit, and geographical region.

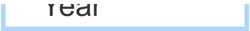
The primary key of a fact table is the composite of all foreign keys referencing dimension tables.

The primary key of a dimension table is a small, meaningless integer. This reduces the size of fact tables, which often contain millions of rows and many foreign keys referencing dimension tables. Since meaningless primary keys never change, the corresponding foreign keys also never change, and the fact table is easy to maintain.

Most data warehouses have many fact tables. Ex: A data warehouse may have a sales revenue fact table and an employee compensation fact table. Usually, different fact tables reference common dimensions, like Date and Location, as well as different dimensions, like Product and Employee.

PARTICIPATION ACTIVITY | 22.5.1: Dimensional design.





Animation content:

Static figure:
Five tables appear.

Tables Employee, Job, Location, and Date have caption dimension tables. Employee has columns EmployeeID, SocialNumber, FirstName, and LastName. EmployeeID is the primary key. Job has columns JobID, Title, JobCode, and LevelCode. JobID is the primary key. Location has columns LocationID, StreetAddress, City, PostalCode, State, and CountryCode. LocationID is the primary key. Date has columns DateID, FullDate, DayNumber, MonthNumber, MonthName, and Year. DateID is the primary key.

Table EmployeeFact has caption fact table. EmployeeFact has columns EmployeeID, JobID, LocationID, HireDateID, BirthDateID, SalaryAmount, BonusAmount, EducationYears, and PerformanceRating. (EmployeeID, JobID, LocationID, HireDateID, BirthDateID) is the primary key.

Five arrows appear. The arrows begin at columns EmployeeID, JobID, LocationID, HireDateID, and BirthDateID of the EmployeeFact table and point to the corresponding dimension table. The arrows from HireDateID and BirthDateID both point to the Date table.

Animation captions:

1. The Employee dimension table includes text that describes employees. SocialNumber contains numeric data but is not quantitative.
2. Additional dimension tables describe corporate jobs, office locations, and dates.
3. Primary keys of dimension tables are meaningless integers.
4. The EmployeeFact table contains numeric data about employees.
5. Foreign keys in EmployeeFact reference dimension tables.
6. A dimension foreign key can appear multiple times in a fact table, with different meanings.
7. The EmployeeFact primary key is the composite of all foreign keys.

PARTICIPATION ACTIVITY 22.5.2: Dimensional design.

1) In a dimensional design for a motor vehicles data warehouse, is the DriverLicenseNumber column in a fact or dimension table?

○ Fact table

○ Dimension table

○ Either fact or dimension table

2) In a dimensional design for a sales data warehouse, is Discount in a fact or dimension table?

○ Fact table

○ Dimension table

○ Either fact or dimension table

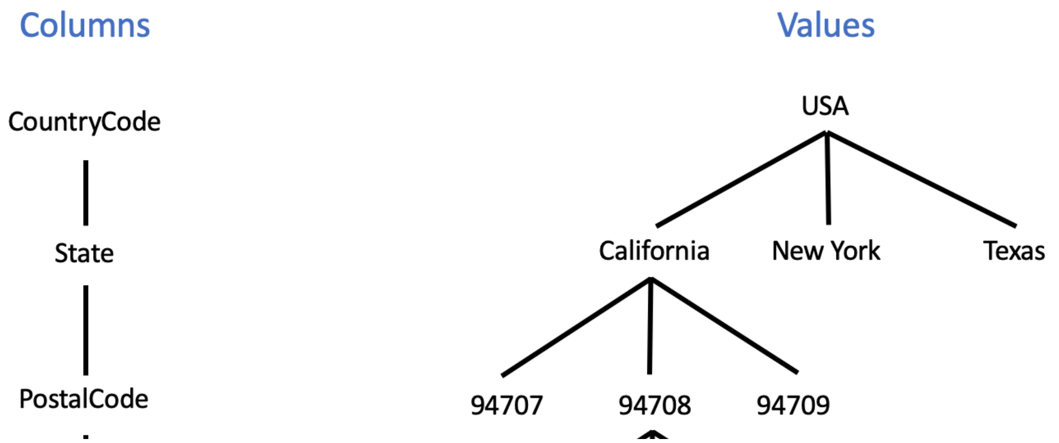
Hierarchies

A ***dimension hierarchy*** is a sequence of columns in which each column has a one-many relationship to the next column. A dimension table usually contains one or more column hierarchies. Ex: The Location table contains CountryCode, State, and PostalCode columns. Each country contains many states, and each state contains many postal codes, so these columns form a hierarchy.

In some cases, several columns are at the same level of a hierarchy. Ex: City and postal code boundaries overlap. Most cities have many postal codes, and some postal codes span multiple cities. City and PostalCode are at the same level of the Location dimension.

Analytic queries usually summarize data at one level of one hierarchy from each dimension. Ex: A query may summarize sales revenue by State (Location dimension), MonthName (Date dimension), and ProductLine (Product dimension). For fast execution, frequently used summary data may be computed in advance and stored in a data warehouse.

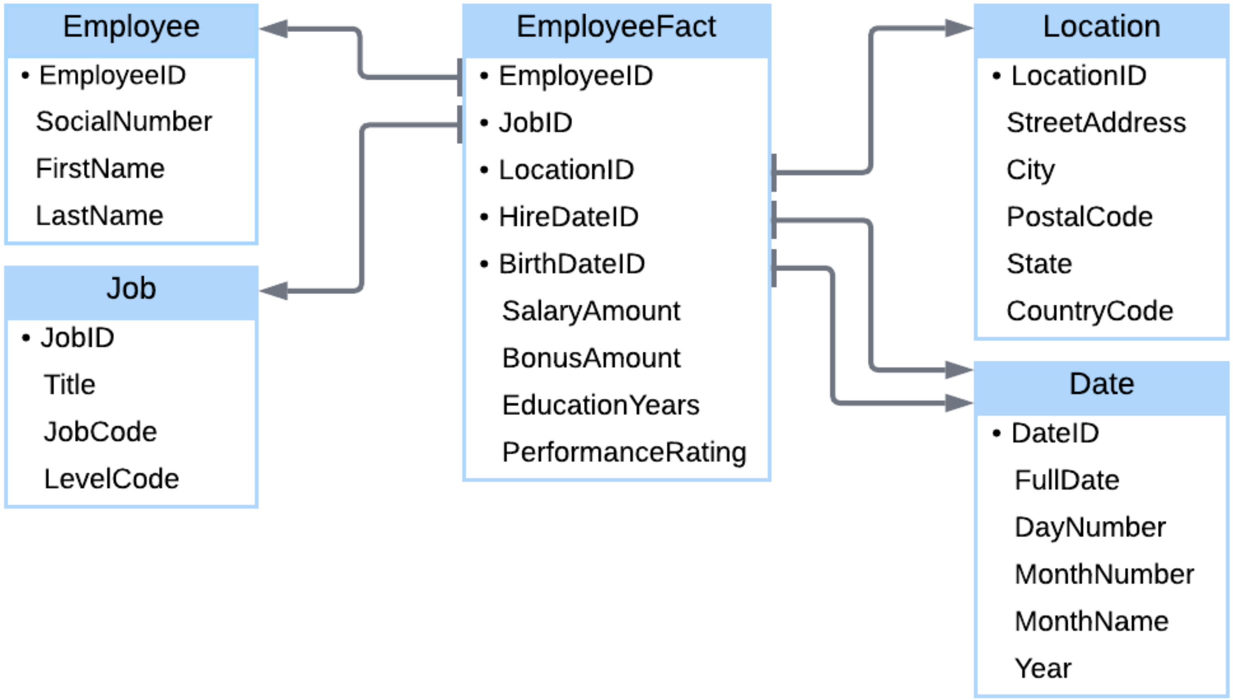
Figure 22.5.1: Location hierarchy.



StreetAddress
22 Oak Street
81 Birch Street
101 Pine Street

PARTICIPATION ACTIVITY 22.5.3: Hierarchies.

Refer to the tables below.



1) In the Job dimension, each job has a unique title and job code, but many jobs are at the same level. What are the hierarchies in the Job dimension?

○ Title and JobCode are at the same level, both below LevelCode.

○ Job dimension has two hierarchies: Title below LevelCode, and JobCode below LevelCode

○ The Job dimension has no hierarchies.

2) In the Date dimension, DayNumber is the day of a month (1 to 31), and FullDate is a specific date ('January 1, 2020'). What is a hierarchy in the Date dimension?

○ Year - MonthNumber - DayNumber

○ Year - MonthName

○ Year - FullDate

3) Which query computes total compensation by country, for employees with job code 220?

○

```
SELECT CountryCode,
SUM(SalaryAmount +
BonusAmount)
FROM EmployeeFact,
Location, Job
WHERE
EmployeeFact.LocationID
= Location.LocationID
AND EmployeeFact.JobID =
Job.JobID
AND JobCode = 220
GROUP BY CountryCode;
```

○

```
SELECT SUM(SalaryAmount
+ BonusAmount)
FROM EmployeeFact,
Location, Job
WHERE
EmployeeFact.LocationID
= Location.LocationID
AND EmployeeFact.JobID =
Job.JobID
AND JobCode = 220
GROUP BY CountryCode;
```

○

```
SELECT CountryCode,
SUM(SalaryAmount +
BonusAmount)
FROM EmployeeFact,
Location, Job
WHERE
EmployeeFact.LocationID
= Location.LocationID
AND EmployeeFact.JobID =
Job.JobID
GROUP BY CountryCode
HAVING JobCode = 220;
```

```
        HAVING JobCode = 110;
```

Date and time dimensions

Since data warehouses track historical data, dimensional designs usually have date and time dimension tables:

- Each row of the ***date dimension*** table corresponds to a day. If an organization tracks data for 100 years, the date dimension contains 36,500 rows (100 years × 365 days per year).
- Each row of the ***time dimension*** table corresponds to a minute of the day. The time dimension contains 1,440 rows (24 hours × 60 minutes per hour).

Fact tables contain foreign keys referencing date, time, or both dimensions, to establish the time of a fact. Ex: The SalesFact table might include a date foreign key to establish date of sale. If an organization tracks time of sale, SalesFact would also include a time foreign key.

The date and time dimensions provide an elegant way to track historical data. Foreign keys StartDateID and EndDateID are added to the fact table and indicate the effective dates of each row. Current rows have an end date in the distant future, such as December 31, 2999. If a fact changes to a new value on date X, the end date of the current row is set to X and a new row is inserted with these values:

- The fact column is the new value.
- StartDateID refers to date X.
- EndDateID refers to December 31, 2999.
- Other columns are identical to the prior row.

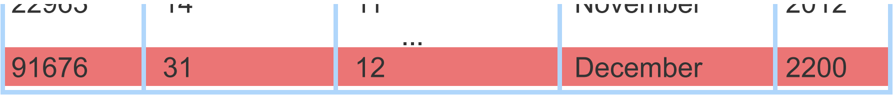
The time dimension is handled in the same way, by adding StartTimeID and EndTimeID foreign keys to the fact table.

Adding start and end foreign keys to the fact table is called ***type 2 design for slowly changing dimensions***. Historical data can be tracked with other designs, but type 2 design is simple, effective, and commonly used.

PARTICIPATION ACTIVITY 22.5.4: Date dimensions.

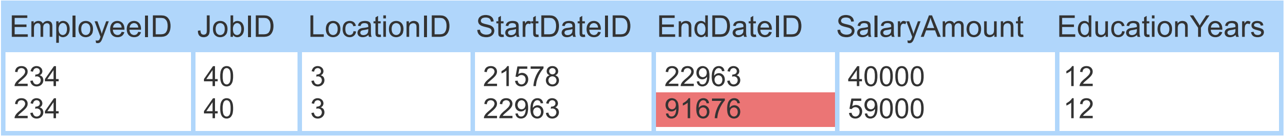
Date

DateID	DayNumber	MonthNumber	MonthName	Year
1	1	1	January	1950
		...		
21578	28	7	July	2009
		...		
22963	14	11	November	2012



22963	14	11	November	2012
		...		
91676	31	12	December	2200

EmployeeFact



EmployeeID	JobID	LocationID	StartDateID	EndDateID	SalaryAmount	EducationYears
234	40	3	21578	22963	40000	12
234	40	3	22963	91676	59000	12

Animation content:

Static figure:
Two tables appear. The Date table has columns DateID, DayNumber, MonthNumber, MonthName, and Year. Date has four rows:
1, 1, 1, January, 1950
21578, 28, 7, July, 2009
22963, 14, 11, November, 2012
91676, 31, 12, December, 2200
Spaces between the five rows indicate that additional rows are not shown. The last row is highlighted.

The EmployeeFact table has columns EmployeeID, JobID, LocationID, StartDateID, EndDateID, SalaryAmount, and EducationYears. EmployeeFact has two rows:
234, 40, 3, 21578, 22963, 40000, 12
234, 40, 3, 22963, 91676, 59000, 12
The value 91676 is highlighted.

Step 1: The Date dimension has one row for each day between January 1, 1950 and December 31, 2200. The Date table appears.

Step 2: EmployeeFact contains historical data. StartDateID and EndDateID indicate effective dates of each row. The EmployeeFact table appears with no rows. Columns StartDateID and EndDateID are highlighted.

Step 3: Between July 28, 2009 and November 14, 2012, employee 234 had a salary of $40,000. The first row appears in EmployeeFact:
234, 40, 3, 21578, 22963, 40000, 12
Value 21578 is highlighted. The corresponding Date row is highlighted. Value 22963 is highlighted. The corresponding Date row is highlighted. Value 40000 is highlighted.

Step 4: On November 14, 2012, the employee received a raise to $59,000. The second row

appears in EmployeeFact:
234, 40, 3, 22963, 91676, 59000, 12
Value 22963 is highlighted. The corresponding Date row is highlighted. Value 59000 is highlighted.

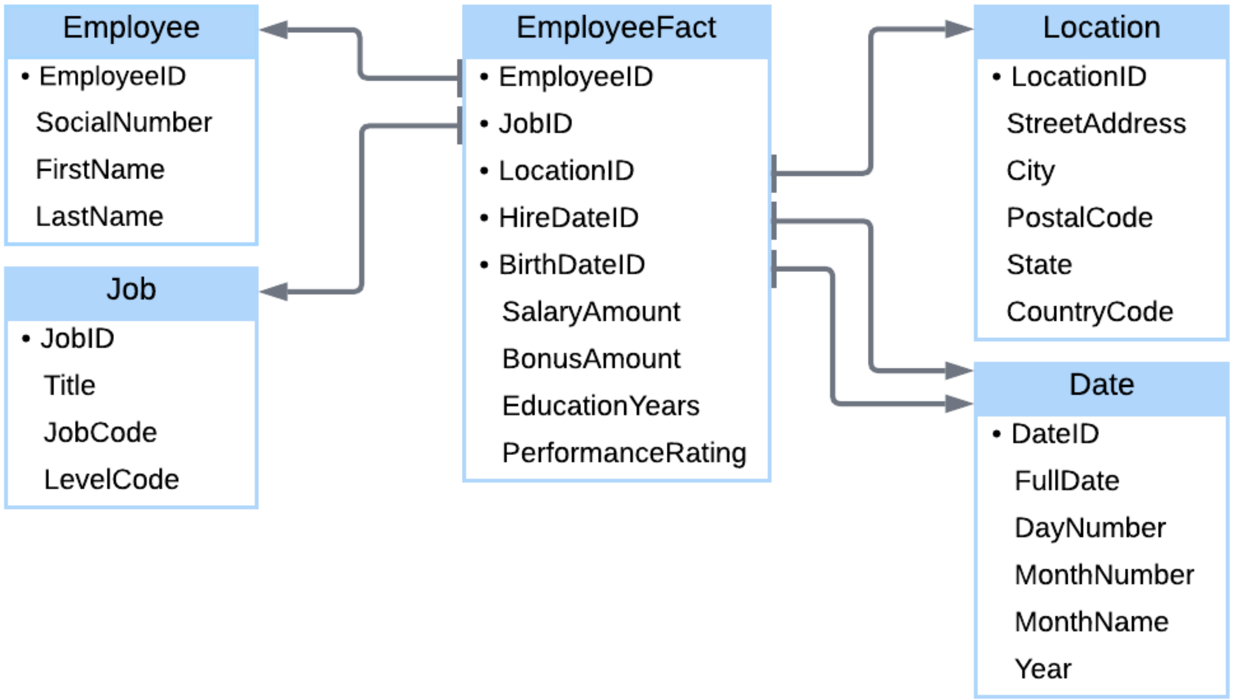
Step 5: The salary of 59,000 is current, so EndDateID refers to a distant future date. Value 91676 in the second EmployeeFact row is highlighted. The corresponding Date row is highlighted.

Animation captions:

1. The Date dimension has one row for each day between January 1, 1950 and December 31, 2200.
2. EmployeeFact contains historical data. StartDateID and EndDateID indicate effective dates of each row.
3. Between July 28, 2009 and November 14, 2012, employee 234 had a salary of $40,000.
4. On November 14, 2012, the employee received a raise to $59,000.
5. The salary of 59,000 is current, so EndDateID refers to a distant future date.

PARTICIPATION ACTIVITY | 22.5.5: Date dimensions.

Refer to the tables below.



The following query reports the average salary for all employees by year and job code:

```
SELECT ___A___, JobCode, AVG(___B___)
FROM Date, Job, Date StartDate, Date EndDate, EmployeeFact
WHERE ___C___ = EmployeeFact.JobID
AND StartDate.DateID = EmployeeFact.StartDateID
AND EndDate.DateID = EmployeeFact.EndDateID
AND Date.Year BETWEEN StartDate.Year AND EndDate.Year
GROUP BY ___D___, JobCode;
```

1) What is identifier A?

Check Show answer

2) What is identifier B?

Check Show answer

3) What is identifier C?

Check Show answer

4) What is identifier D?

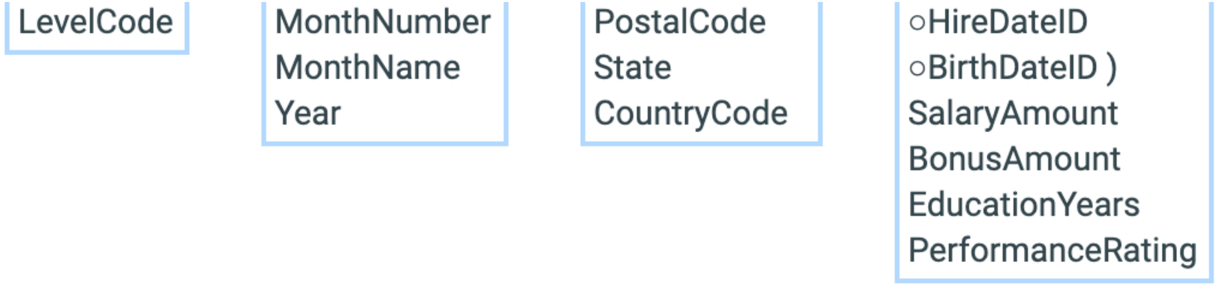
Check Show answer

CHALLENGE ACTIVITY 22.5.1: Data warehouse design.

544874.3500394.qx3zqy7

Start

Job	Date	Location	EmployeeFact
●JobID	●DateID	●LocationID	● (○EmployeeID
Title	FullDate	StreetAddress	○JobID
JobCode	DayNumber	City	○LocationID



The following query reports the city with the highest salary amount for employees with job code 3. Complete the missing values.

```
SELECT City, MAX(_____(A)_____)
FROM _____(B)_____, Location, Job
WHERE EmployeeFact.LocationID = Location.LocationID
AND EmployeeFact.JobID = _____(C)_____
AND _____(D)_____ = 3
GROUP _____(E)_____ _____(F)_____;
```

(A) Ex: Identifier

(B)

(C)

(D)

(E)

(F)

1 2

Check Next

Exploring further:

- The Data Warehouse Toolkit
- Slowly changing dimensions

22.6 Other database architectures

In-memory databases

An ***in-memory database*** is a database that stores data in main memory, instead of or in addition to storage media. Main memory is much faster than storage media, such as flash memory and disk drives. Consequently, in-memory databases are appropriate for analytic applications, which require fast execution of lengthy queries. In-memory databases are also appropriate for applications that rapidly insert high volumes of data, such as data collection from internet devices.

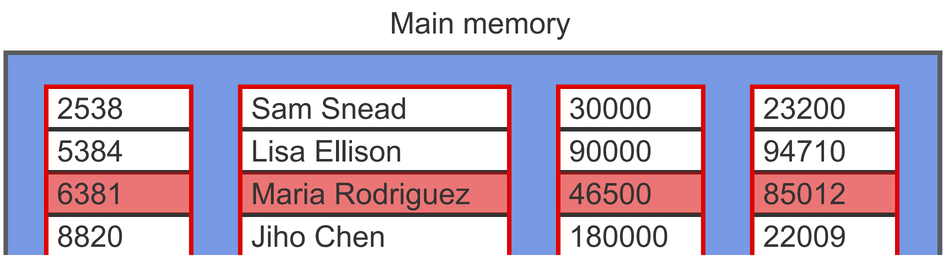
Historically, main memory cost per byte was too high and capacity was too limited for most database applications. In the past decade, however, main memory cost has dropped, and capacity has increased significantly. In-memory databases can now store terabytes of data, which is adequate for many databases.

Main memory is volatile and lost when power fails or the database process crashes, so in-memory data is periodically backed up on storage media. In-memory databases may also record insert, update, and delete operations in a log file on storage media, which can be used to reconstruct databases in the event of a crash.

Many databases now allow database administrators to store selected tables in memory while other tables remain on storage media:

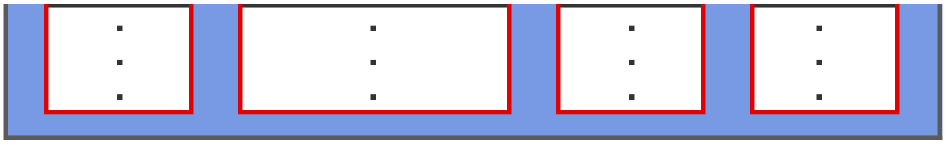
- *SQL Server In-Memory OLTP* is an extension to SQL Server supporting in-memory tables. In-memory tables offer the same transaction and recovery options as storage media tables.
- *Oracle Database In-Memory* creates in-memory copies of tables. The table source data remains on storage media, grouped by rows in blocks. In-memory copies are physically organized by column, rather than by row. The memory's columnar organization is optimal for analytic queries, which often summarize large volumes of data from one or two columns.
- *MySQL* assigns a specific storage engine to individual tables. Both the *MEMORY* and *MySQL NDB Cluster* storage engines support in-memory tables. MEMORY does not support transactions or recovery in the event of a failure, and consequently is appropriate for temporary tables only. NDB Cluster supports transactions, recovery, and distributed data, and is recommended for persistent data.

PARTICIPATION ACTIVITY 22.6.1: Oracle database in-memory.

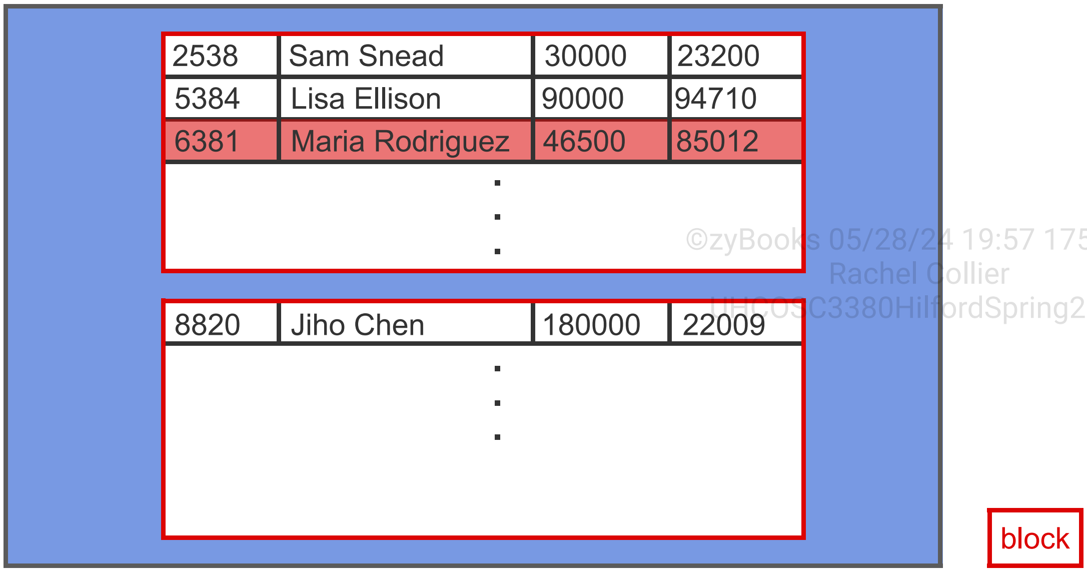


Main memory

2538	Sam Snead	30000	23200
5384	Lisa Ellison	90000	94710
6381	Maria Rodriguez	46500	85012
8820	Jiho Chen	180000	22009



Storage Media



2538	Sam Snead	30000	23200
5384	Lisa Ellison	90000	94710
6381	Maria Rodriguez	46500	85012

8820	Jiho Chen	180000	22009

Animation content:

Static figure:
Two diagrams appear with captions main memory and storage media.

The storage media diagram contains data for employee number, employee name, salary, and zip code. Data is organized in two blocks, each containing complete rows of data . The first block contains three rows:
2538, Sam Snead, 30000, 23200
5384, LIsa Ellison, 90000, 94710
6381, Maria Rodriguez, 46500, 85012
The second block contains one row:
8820, Jiho Chen, 180000, 22009

The main memory diagram has the same data. Data is organized in four blocks, each containing the data in a single column.

Step 1: Oracle database organizes tables on storage media by row. The storage media diagram appears without the Maria Rodriguez row.

Step 2: Oracle Database In-Memory creates a copy of a table in memory. The main memory diagram appears without the Maria Rodriguez data.

Step 3: In-memory data is organized by column to optimize for analytic queries. The four blocks of column data in main memory are highlighted.

Step 4: Copies of tables are synchronized. Maria Rodriguez data is added to each of the four main memory blocks. The Maria Rodriguez row is added to the first storage media block.

Animation captions:

1. Oracle database organizes tables on storage media by row.
2. Oracle Database In-Memory creates a copy of a table in memory.
3. In-memory data is organized by column to optimize for analytic queries.
4. Copies of tables are synchronized.

PARTICIPATION ACTIVITY | 22.6.2: In-memory databases.

Indicate whether in-memory databases always, sometimes, or never support each of the following characteristics.

1) Transactions

○ Always

○ Sometimes

○ Never

2) Recovery

○ Always

○ Sometimes

○ Never

3) Fast query execution

○ Always

○ Sometimes

○ Never

Embedded databases

An ***embedded database***, sometimes called an ***in-process database***, is a database that is packaged with a programming language. An embedded database and application program execute together in a single software process. Embedded databases are used in single-user applications that require no database administration, such as applications designed for mobile devices.

A software process containing an embedded database runs on a single hardware tier but is often part of a complex multi-tier system. Ex: A JavaScript program running within a web browser may contain an embedded database to manage local data. The program may connect to a web server, application, and services running in lower tiers of a multi-tier architecture.

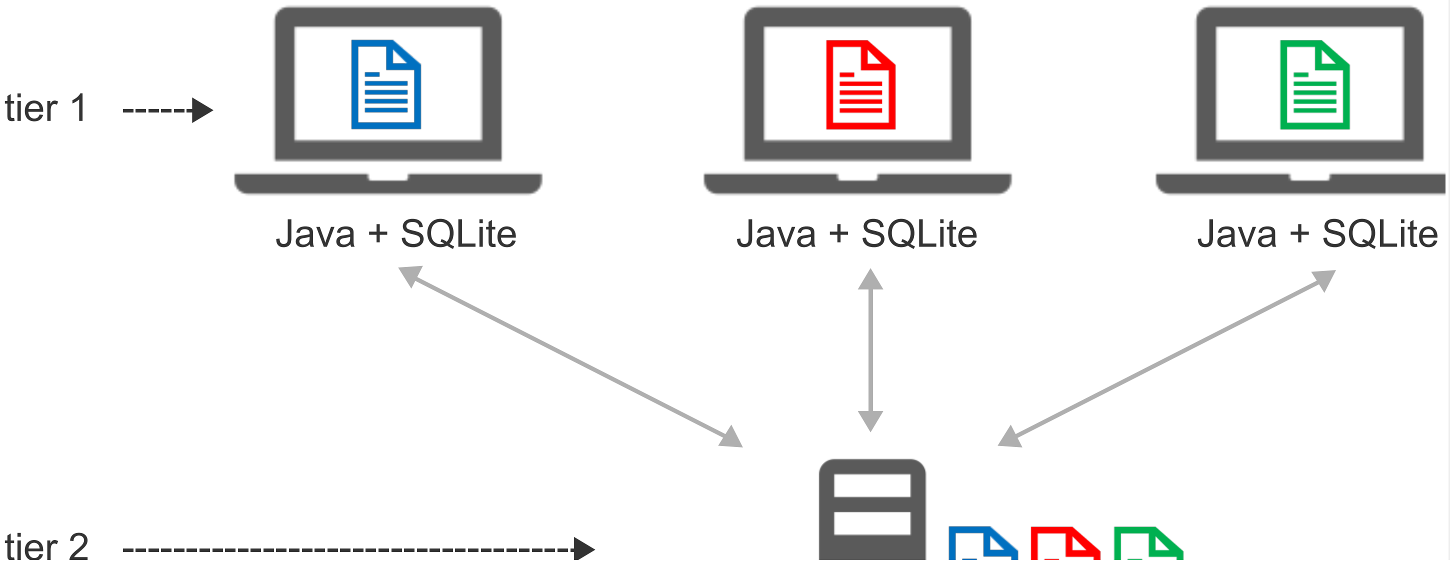
SQLite is the dominant embedded relational database. SQLite is an open source relational database and supports all major programming languages.

Several leading embedded relational database products have been deprecated due to limited revenue opportunity and the dominance of SQLite. ***SQL Server Compact*** is an embedded database from Microsoft. The last major release of SQL Server Compact was in 2011, and Microsoft will discontinue support after 2021. The MySQL software library ***libmysqld*** configures MySQL as an embedded database but was discontinued as of MySQL release 8.0.

Terminology

*An **embedded database** is not the same as **embedded SQL**. Embedded database is a database architecture, while embedded SQL, described elsewhere in this material, is a database programming technique.*

PARTICIPATION ACTIVITY 22.6.3: Embedded database in a two-tier architecture.



Animation content:

Static figure:
The diagram is arranged in two tiers with captions tier 1 and tier 2. Tier 1 contains three laptop computers with captions Java + SQLite. Each computer displays one document. Each document is a different color. Tier 2 contains a server computer with caption Oracle Database. Next to the server computer are three documents in the same colors as tier 1 documents. Double-headed arrows are between each laptop computer and the server computer.

Animation captions:

1. Students take and edit notes on laptop computers.
2. A Java application stores notes in SQLite on the laptop.
3. Students can upload notes to a server and share with others.
4. Thousands of students may access the server concurrently. Oracle Database scales better than SQLite and is installed on the server.
5. The application has a two-tier architecture.

PARTICIPATION ACTIVITY | 22.6.4: Embedded database.

What are the benefits of embedded databases?

1) Extensive administration utilities

○ True

○ False

2) Ease of programming

○ True

○ False

3) Low memory and processing requirements

○ True

○ True

○ False

4) Scales to large numbers of users and database sizes

○ True

○ False

Federated databases

A ***federated database*** is a collection of two or more participating databases underneath a coordinating software layer. The participating databases are autonomous and heterogeneous:

- An ***autonomous database*** operates independently of other participating databases. An autonomous database is administered and can be queried as if the database were not part of a federated database.
- ***Heterogeneous databases*** either run under different database systems or have incompatible schema. Databases with incompatible schema might have inconsistent primary and foreign keys, similar tables with different designs, or similar columns with different names and data types

The coordinating software layer is called ***middleware***, since the software lies between application programs and database software. Many federated database middleware products are available. Ex: InfoSphere Federated Server from IBM and WebLogic Server from Oracle. Although product capabilities vary, most products have the following components:

- A ***global catalog*** is a directory of participating database objects, such as tables, columns, and indexes.
- A *global query processor* decomposes a federated query into queries for each participating database.
- A ***database wrapper*** converts the decomposed queries to the appropriate syntax for each participating database.

Some products support ***SQL/Management of External Data***, or ***SQL/MED***, an extension of the SQL standard for federated databases. SQL/MED adds constructs such as nicknames and user mappings to SQL. A ***nickname*** is a federated database name for a participating database object, such as tables and columns. A ***user mapping*** associates a federated database user with a participating database user.

Figure 22.6.1: SQL/MED nickname and user mapping examples.

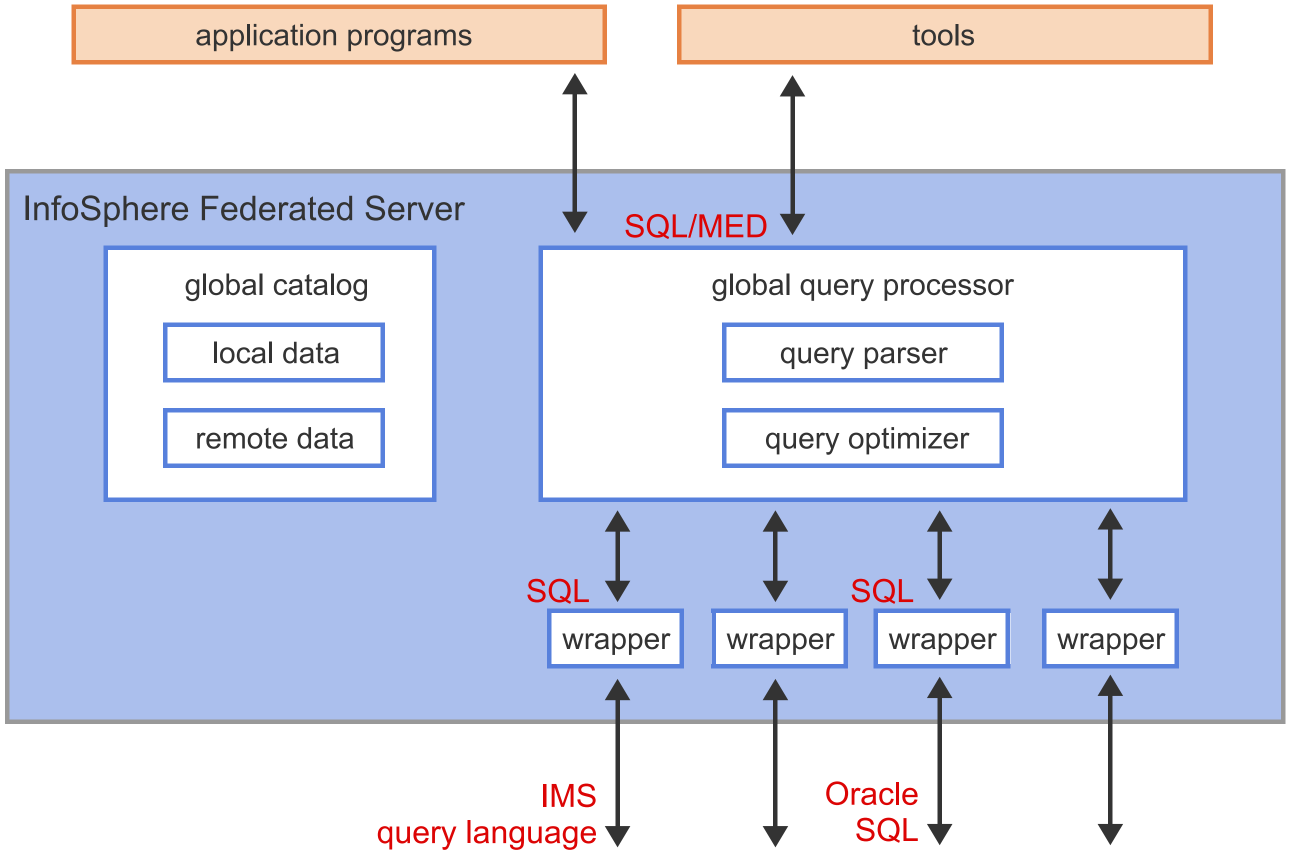
```
CREATE NICKNAME Employee
FOR DB2SERVER.HRdatabase.Emp2table;

CREATE USER MAPPING FOR SamSnead
SERVER DB2SERVER
OPTIONS (REMOTE_AUTHID 'sam.snead@gmail.com', REMOTE_PASSWORD
'X!8sflHn');
```

From the perspective of users and application programs, federated databases are more complex than distributed databases and data warehouses. In a distributed database, the assignment of data to nodes is invisible to users and programs. In a data warehouse, the data sources are invisible. In a federated database, however, the participating databases may be visible. Ex: Incompatible schema may specify different update rules for similar foreign keys, so the behavior of foreign key updates depends on the participating database.

Although a federated database does not provide a seamless view of data, a federated database is relatively easy to build and often the only practical way to combine data from existing, incompatible databases.

PARTICIPATION ACTIVITY | 22.6.5: InfoSphere Federated Server.



Animation content:

Static figure:
A rectangle with captain InfoSphere Federated Server contains a global catalog, a global query processor, and four wrappers. The global catalog contains local data and remote data. The global query processor contains a query processor and a query optimizer.

Rectangles with captions application programs and tools appear above InfoSphere Federated Server. Rectangles with captions IMS, DB2, Oracle, and Excel appear below InfoSphere Federated Server.

Double-headed arrows with caption SQL/MED connect application programs and tools to the global query processor. Double-headed arrows with caption SQL connect the global query processor to each wrapper. Double-headed arrows connect each wrapper to IMS, DB2, Oracle, and Excel. The caption IMS query language appears next to the arrow pointing to IMS. The caption Oracle SQL appears next to the arrow pointing to Oracle.

Animation captions:

1. InfoSphere Federated Server is a federated database middleware product from IBM.
2. InfoSphere supports relational and non-relational databases. IMS is an older, non-relational database from IBM.
3. The global catalog contains local data about internal InfoSphere objects and remote data about participating database objects.
4. The global query processor decomposes a SQL/MED query into SQL queries for each database.
5. Database wrappers translate standard SQL to database query language.
6. The query parser checks SQL/MED syntax and converts queries into an internal format.
7. The query optimizer determines an optimal execution plan using information in the global catalog.

PARTICIPATION ACTIVITY | 22.6.6: Federated databases.

The following questions refer to InfoSphere Federated Server.

1) Which component merges query

results from participating databases?

○ Wrapper

○ Global query processor

○ Global catalog

2) Which component determines the execution plan for a decomposed query?

○ Database

○ Wrapper

○ Global query processor

3) Is referential integrity maintained when a federated query inserts, updates, or deletes primary and foreign keys?

○ Always

○ Sometimes

○ Never

Data lakes

A ***data lake*** is an analytic database of raw, unprocessed data copied from multiple data sources. Data lakes share some characteristics of data warehouses and some characteristics of federated databases:

- Like a data warehouse, a data lake is a separate database designed for analytic queries and consisting of data extracted from multiple source systems.
- Like a federated database, data in a data lake is not cleansed, integrated, or restructured. Data is stored in the original format and structure. Depending on the data source, data may be loaded continuously rather than at fixed intervals.

Data lakes often contain large volumes of data, such as sensor data or website clicks. Data lakes also contain unstructured data, such as images, video, and text documents, which consume megabytes or gigabytes per data item. As a result, data lakes usually require a large amount of storage and utilize inexpensive, but relatively slow, storage media.

Data lakes emerged in response to the high cost of building and maintaining data warehouses. Since data is copied exactly as stored in the source system, constructing a data lake is relatively simple. On the other hand, formulating and understanding queries is more difficult with a data lake than a data warehouse. Consequently, data lakes are more suitable for data scientists, who are trained to work with complex, unstructured data, than for business analysts.

Ex: Healthcare organizations have had mixed success with data warehouses. Healthcare data, such as doctor's notes and clinical data, is often unstructured, complex, and difficult to analyze. A data lake supported by specialized data scientists is a better solution for many healthcare organizations.

Figure 22.6.2: Data lake vs. data warehouse.

	Data lake	Data warehouse
Consumer	Data scientist	Business analyst
Scope	Broad	Selective
Data types	Structured and unstructured	Structured
Data quality	Raw	Cleansed
Database design	Identical to source	Dimensional
Storage media	High capacity	High speed
Effort to build/maintain	Low	High
Effort to query/analyze	High	Low

PARTICIPATION ACTIVITY | 22.6.7: Database architectures.

Each scenario describes particular database requirements. Match the database architecture to the scenario.

If unable to drag and drop, refresh the page.

Embedded database **Federated database** **Data warehouse** **Data lake**

A university maintains student records, catalog information, and faculty profiles in incompatible operational databases. The university wants to simplify interaction with these databases for the registrar's office.

An e-commerce company wants to understand why online customers are adding products to shopping carts but not completing purchases. The company will analyze web click patterns along with customer data that is publicly available on social media.

A manufacturing company wants to 'slice and dice' quarterly sales data to analyze manufacturing cost and sales trends of product lines in different regions.

One software layer in a complex web architecture needs to maintain session data. The data is complex but local to the software layer and is not permanently stored.

Reset

CHALLENGE ACTIVITY | 22.6.1: Other database architectures.

544874.3500394.qx3zqy7

Start

Each scenario describes particular database requirements. Match the database architectu

Pick (a) A non-profit organization stores a user's demographics via th built-in cache.

Pick

(b) A non-profit organization stores info on volunteers, donors, a separately maintained databases. The organization wants to sin with these databases for their legal team.

Pick

(c) A company wants to analyze how to increase production effi down material costs and hours of labor at each stage of produc

Pick

(d) A company wants to analyze the impact of news stories on a across Europe and Asia.

1

Check

Try again

Exploring further:

- Oracle Database In-Memory
- SQLite embedded database
- InfoSphere Federated Server