

IS475/675 Agenda: Week 15

Monday

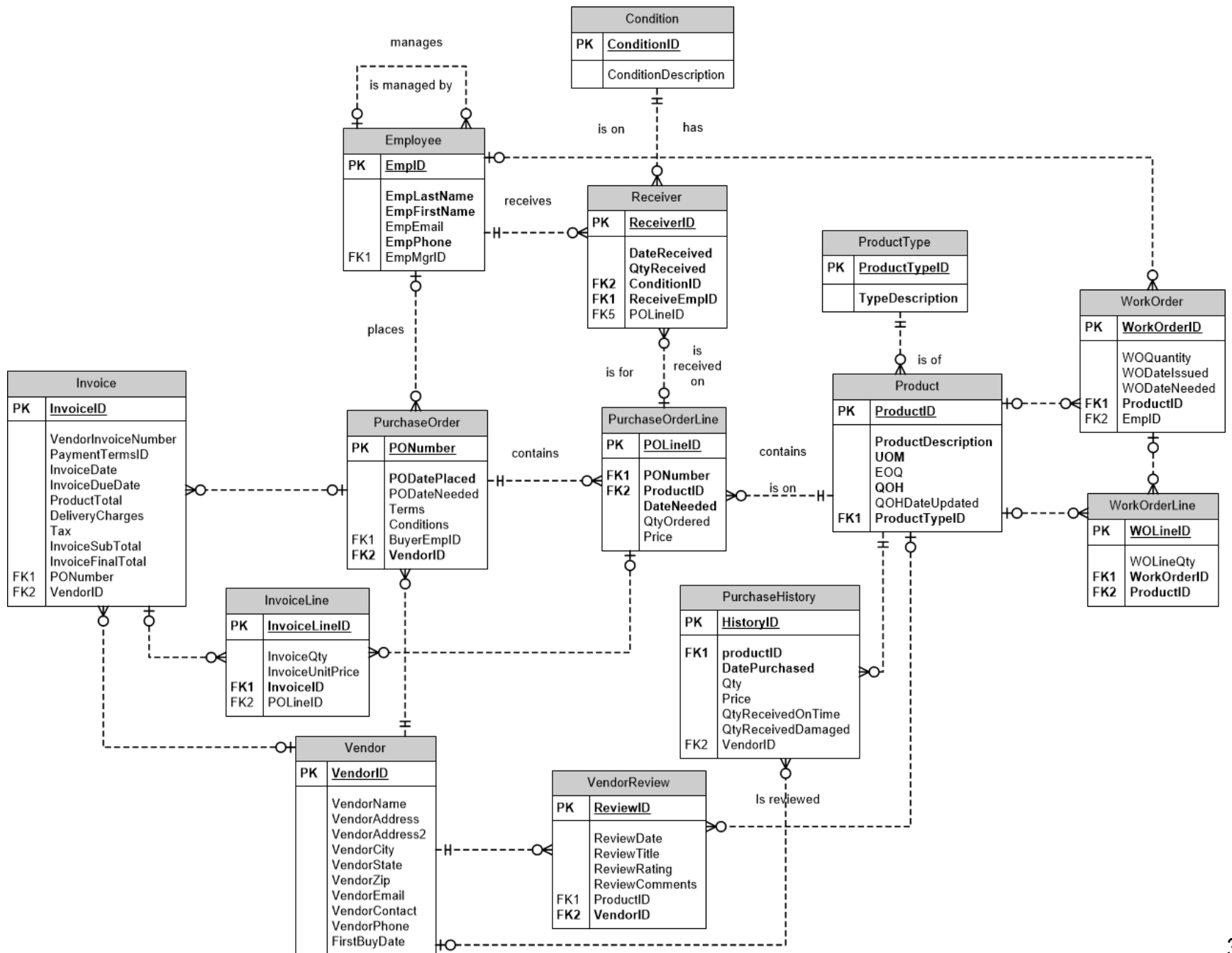
- Describe HW#10.
- Finish discussing how a DBMS supports partitioned applications.
- Review/refresh physical database design.
- Discuss how a DBMS and a database designer can optimize the efficiency of a database.

Wednesday

- Identify the different databases in an organization
- Present how data is moved among databases.
- Introduce organizational data architecture.

HW#10

- Before starting on the assignment, complete SQL lab exercise 9.
- Add new data to the Product table. Don't drop and rebuild the table – figure out how to add new data to the table.
- Create and populate four new tables in the Mountain Design Purchase Order Database.
- Write queries – SELECT statements.
- For the sample queries, turn in all SQL code, including all code that composes views and CTE's.
- Turn in all SQL code in a file that can be copied and pasted to SQL Server. Do not turn in “pictures” of your SQL code.
- Be sure to include the names of the team on the document – also indicate the name of the database where we should test your work.



What have we been discussing?

- Partitioned (also called “layered”) applications
- Database programming to support partitioned applications:
 - Functions, stored procedures, and triggers
- Features of a database management system to support partitioned applications:
 - Concurrency control
 - Security
 - Database backup and recovery

Database security

- Database Security:
Protection of the data
against accidental or
intentional loss,
destruction, or misuse.
Threats to database
security include the list to
the right.
- Accidental losses attributable to:
 - Human error.
 - Software failure.
 - Hardware failure.
- Theft and fraud.
- Improper data access:
 - Loss of privacy (personal data).
 - Loss of confidentiality (corporate data).
- Loss of data integrity.
- Loss of availability.

Top Database Security Threats

- **Privilege abuse:** excessive access privileges, using privileges to create unauthorized data sets, enhancing privilege access from user to administrator.
- **SQL Injection:** adding SQL code via web forms (or other vulnerable input channels) to execute fraudulent commands.
- **Lack of encryption** of identifying data.
- Weak authentication.
- Backup data availability.
- Weak audit trail.

SQL Injection

```
SELECT email, passwd, login_id, full_name  
FROM    members  
WHERE email = '$email'
```

So, now imagine that someone enters something like this for the email address:

Anything' or 'x' = 'x

```
SELECT email, passwd, login_id, full_name  
FROM    members  
WHERE email = 'Anything' or 'x' = 'x'
```

DBMS security features

- Views (frequently referred to as subschemas).
- Integrity controls.
- Authorization rules.
 - Controls incorporated in the DBMS.
 - Restrict access to specific data.
 - Restrict actions that can be taken.
- User-defined procedures.
 - Trigger an authorization procedure which asks additional identification questions.
 - Written in a standard programming language or proprietary language.
- Stored procedures for all data input/access to the database.
- Encryption.
- Authentication schemes.
 - Biometric devices.

SQL statements used for security

SQL Statement	Action
CREATE USER	Allows the DBA to create a new user.
GRANT	Allows the user to give other users privileges to access the user's objects.
CREATE ROLE	Allows the DBA to create a collection of privileges that can be assigned as a group.
ALTER USER	Allows users to change their passwords. Can also be used to change other attributes of a user.
REVOKE	Removes privileges on an object from a user, users, or role.

Pop Quiz!

Which of the following statements regarding database security is TRUE?

- a. A database management system does not have any methods of ensuring appropriate access to data in the database.
- b. A view is not used to control access to tables in a database.
- c. All data stored in a database should be encrypted.
- d. Some database management systems provide a way to encrypt data.

Which of the following database security measures can help protect against SQL injection attacks?

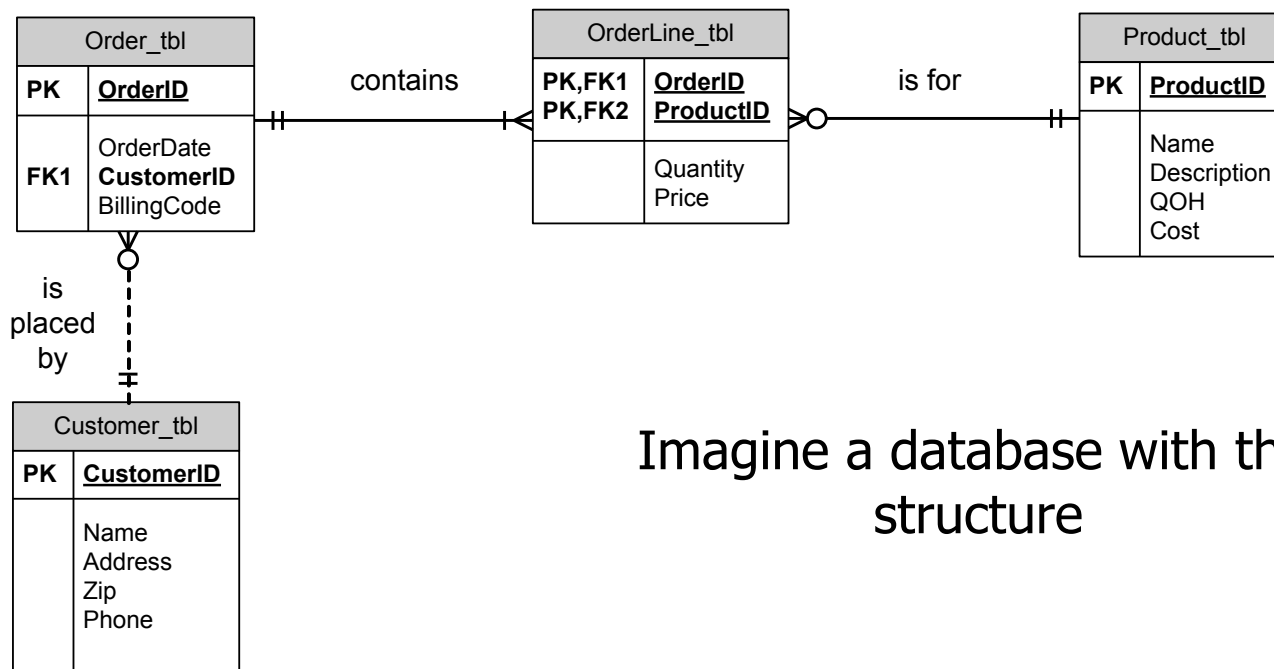
- a. Using authorization rules to prevent access to a database.
- b. Using database constraints such as referential integrity.
- c. Using stored procedures to check data input fields to see if they contain SQL code.
- d. Using views to prevent programmer access directly to database tables.

Database backup and recovery

- What is backup and recovery?
 - **Backup:** A method of storing data from a database in a format that can be used to rebuild the database very quickly if necessary to rebuild.
 - **Recovery:** Mechanisms for restoring a database very quickly and accurately after loss or damage.
- Why have backup and recovery?
 - Human error.
 - Hardware failure.
 - Incorrect or invalid data.
 - Program errors.
 - Viruses.
 - Natural catastrophes.

Backup and recovery are based on transactions

- A transaction is one or more database actions (SQL statements) that are treated as a single unit of work.
 - If the transaction is successful, then the transaction is **committed**.
 - If the transaction is not successful, then the transaction is **rolled back** or **aborted**.



Imagine a database with this structure

Accepting an order for a product

```
INSERT INTO order_tbl VALUES  
  (123, '28-apr-2025', 765, 'net30');
```

```
INSERT INTO orderline_tbl VALUES  
  (123, 6812, 10, 34.99);
```

```
UPDATE    product_tbl  
SET       qoh = qoh - 10  
WHERE     prod_no = 6812;
```

A transaction is:



Atomic: The transaction cannot be sub-divided; it must all be completed and saved to disk for it to be valid.



Consistent: The rules and constraints of the database are consistent for any data written to disk.



Isolated: Transactions act like they are running separately from any other transactions.



Durable: Changes to disk are permanent and require another transaction to occur for any modifications to happen.

DBMS's have methods to control transactions

- Databases that support transactions provide specific commands for starting, committing, and rolling back transactions.
 - Begin transaction.
 - End transaction.
 - Commit.
 - Rollback.
 - Autocommit.

Database backup is conducted in 3 processes

Backup: A DBMS software utility provides a way to do a complete, full or incremental backup of the database in a consistent state.

- Complete: entire database copied at pre-specified times. Not real-time.
- Incremental: rows that have changed since the last full backup.
- Mirror image: A copy of the database stored in a separate location on a separate device that is updated in real-time.

Journalize: A DBMS software utility provides an audit trail of changes to the database.

- Transaction log: contains all data used to process changes against the database. Maintained real-time.
- Database change log: contains a before-image and an after-image of each row modified by a database transaction. Maintained real-time.

Checkpoint: A DBMS software utility that periodically suspends all transaction processing and synchronizes files within the database.

- Some databases, such as Oracle, do not actually halt processing. They simply write checkpoint information to physical files.
- The purpose of a checkpoint is to minimize the amount of time it takes to restore a system.

Recovery methods

- A DBMS has a utility to recover the database. Usually referred to as the Recovery Manager.
- The method of recovery depends on the type of failure.
- Recovery Manager usually has the following options:
 - **Switch:** Switches to a replica of the database on a different storage device.
 - Requires that a **mirror image** of the database is stored.
 - Can be expensive.
 - Assumption is a storage failure, not a failure in transaction integrity, occurred.
 - **Restore/Rerun:** Reprocesses the transactions for a given time period against a correct version of the database.
 - Assumption is that a failure in transaction integrity has occurred.
 - Can be very time-consuming.

Issues in database backup and recovery

- Cost.
 - Media.
 - Computer overhead (processor, memory, disk) to create journalizing files, control files, checkpoint files, etc.
 - Personnel to supervise and tune.
- Time.
 - Can result in regularly scheduled downtime.
 - Can make the system slower.

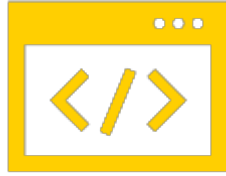
Pop Quiz!

A discrete unit of work that must be processed completely or not at all when using a database management system is called a:

- a. before image.
- b. journalizing facility.
- c. ERD.
- d. transaction.
- e. subschema

A large, global organization such as FedEx, that requires 24 hours a day/ 7 days a week access to its database, would probably choose which of the following recovery methods?

- a. rollback.
- b. restore/rerun.
- c. switch.
- d. checkpoint utility.
- e. transaction authorization.



Logical design

Process of translating user system specifications into a database design.

System specifications can come from existing forms, spreadsheets, and user narratives.

System specifications are gathered from the client – the people who will ultimately use the database to support their work.



Physical design

The process of translating a logical description of data into technical specifications for storing and retrieving data.

Preparing documentation for actual implementation of tables in a database.

Relatively little communication between designer and ultimate user when doing physical design.

Database Design Goals

Logical Design

- Ensure that all data necessary for transaction processing and decision-making is stored and accessible.
- Protect the integrity of the data.
- Provide for change.

Physical Design

- Maintain the integrity of the data.
- Enhance performance of access to data.
- Simplify access to data.

Tasks in physical design

Convert “logical” entities into “physical” tables.

- Identify all necessary data attributes.
- Determine correct size and data type for each data attribute so that it can be a physical field.
- Choose an appropriate primary key.
- Identify foreign keys necessary to sustain relationships.
- Define necessary constraints.

Enhance performance

- Identify size and access methods of data.
- Choose appropriate hardware.
- Create indices.
- De-normalize the design as necessary.
- Partition the data as necessary.
- Create design and procedures for archiving data.

Choosing datatypes for fields

- A **datatype** is a name or label for a set of values and some operations which one can perform on that set of values.
- Examples in SQL: varchar, date, number, money
- SQL is “strongly data typed.” What does that mean?
- Objectives for choosing an appropriate data type:
 - Minimize storage space.
 - Represent all possible values.
 - Improve data integrity.
 - Support all data manipulations.

Controlling data integrity with rules for fields

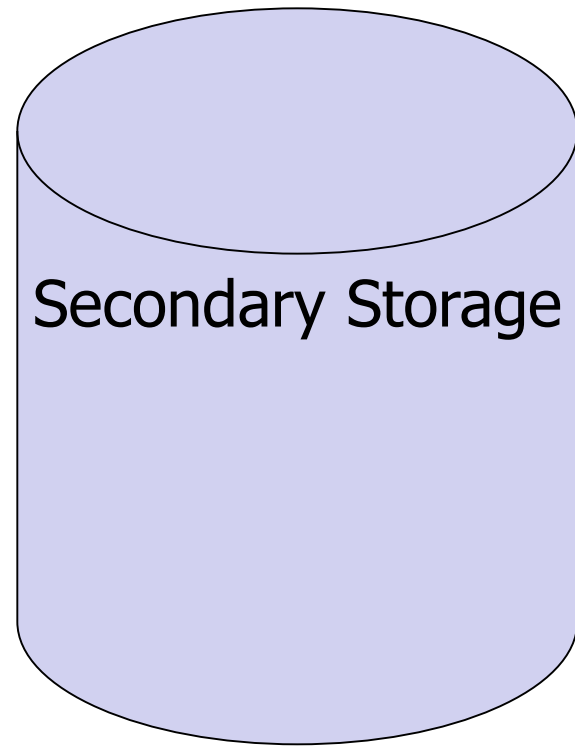
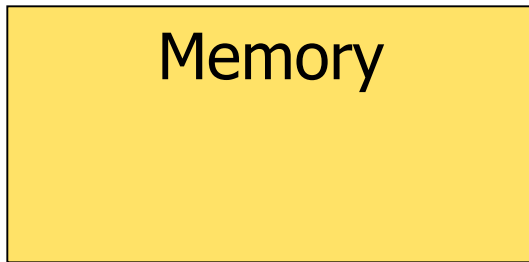
- Default value
- Range control – CHECK constraint
- Null value control
- Primary key constraints
- Referential integrity

Choosing an appropriate primary key

- General rules:
 - Must be a unique value for each row in the table.
 - Cannot be null.
 - Should be static over the life of the row.
- Physical primary key design heuristics:
 - Should be a single attribute.
 - Should be a numeric data type if possible. Should never be a VARCHAR data type.
 - Should not be “intelligent.”

Overview of Database Performance

- Key metrics for database performance
 - Minimize response time to access data in a database.
 - Minimize response time to change contents in a database.
- Most concerned with balancing disk access time and memory capacity.
 - Relative speed of disk access vs main memory processing?
 - Relative size of disk vs. size of main memory?



Ways to Improve performance

1. By optimizing use of existing resources.
2. By using better or more resources.
3. By creating indexes.
4. By denormalizing the database.
5. By partitioning the database.
6. By storing derived data.
7. By creating procedures to differentiate and manage data.

- Add or change resources to improve performance.
 - Will help a little: more processor power.
 - Will help more: more memory.
 - Will really help: Faster, more efficient disk or a fully in-memory database.
https://en.wikipedia.org/wiki/Hard_disk_drive_performance_characteristics
 - https://en.wikipedia.org/wiki/In-memory_database
 - <https://aws.amazon.com/nosql/in-memory/>
- Solid-state drives/arrays
- RAID: Redundant arrays of inexpensive (or independent) disks.
 - A set of multiple physical disk drives that appear to the designer and user as a single storage unit.
 - Segments of data, called stripes, cut across all of the disk drives.
 - Access can occur concurrently.
 - <http://www.acnc.com/raid>

Which of the following statements is **TRUE** about processing data on a computer with a DBMS?

- a) The speed of processing data that is in main memory is significantly slower than accessing data from disk.
- b) Processing data with a DBMS can be greatly enhanced by increasing the processing speed of the computer.
- c) A major bottleneck in processing data with a DBMS is the time it takes to access data on a secondary storage device (like a disk).
- d) A DBMS usually accesses (reads/writes) one logical record (a row) of a table at a time from disk.
- e) All of the above statements are TRUE.

Pop Quiz!

When we say that SQL is strongly data-typed it means:

- a) That all data types are available in SQL.
- b) That only one data type should be used for a field that is a primary key.
- c) That a field can have more than one data type.
- d) That a field's data type can enforce error-checking and help preserve the integrity of data.
- e) That all fields should be a VARCHAR data type of a maximum size of 255 characters – VARCHAR(255).

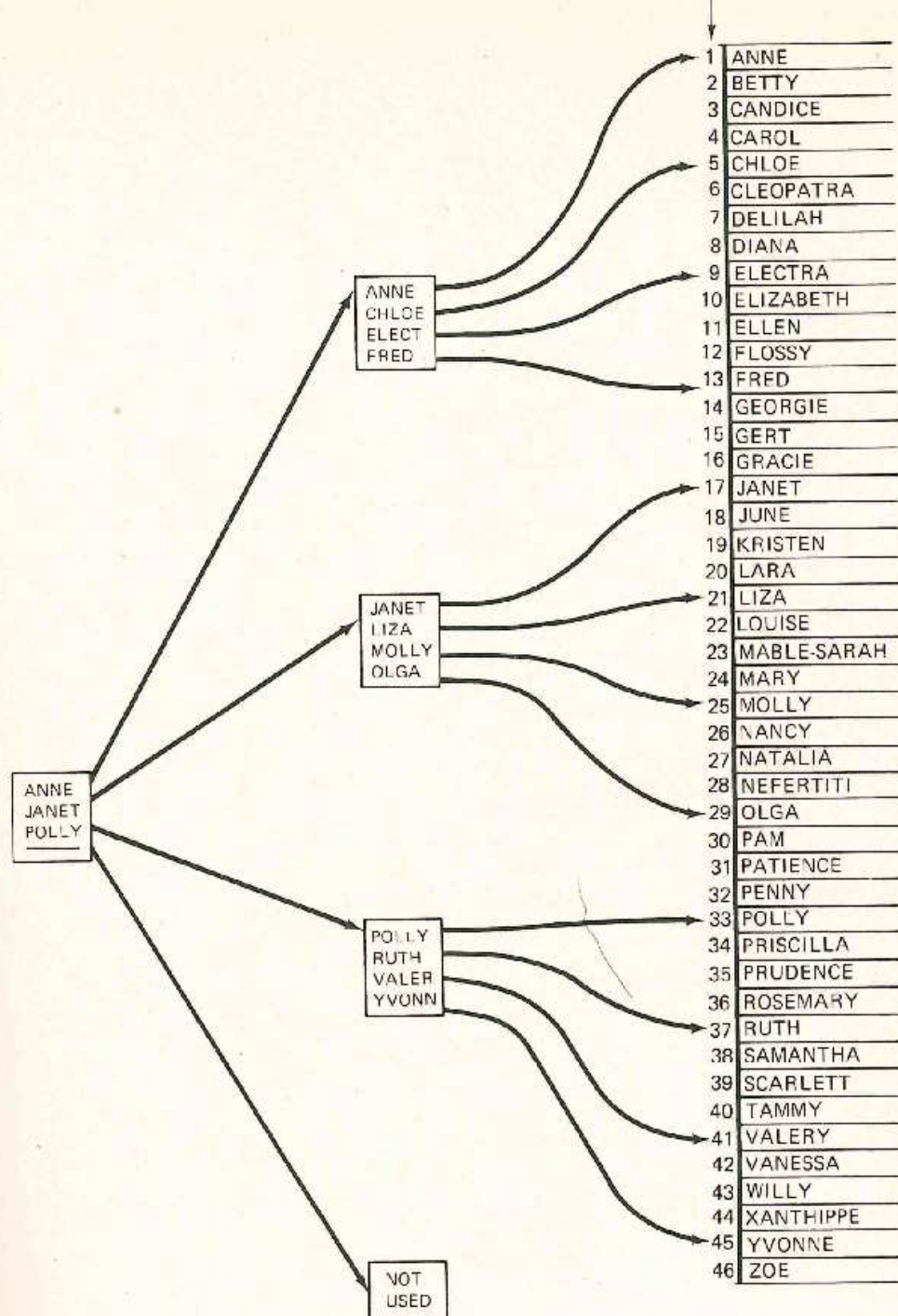
Indexes are the single most important tool a database **programmer and/or administrator** can use to improve the performance of a database.

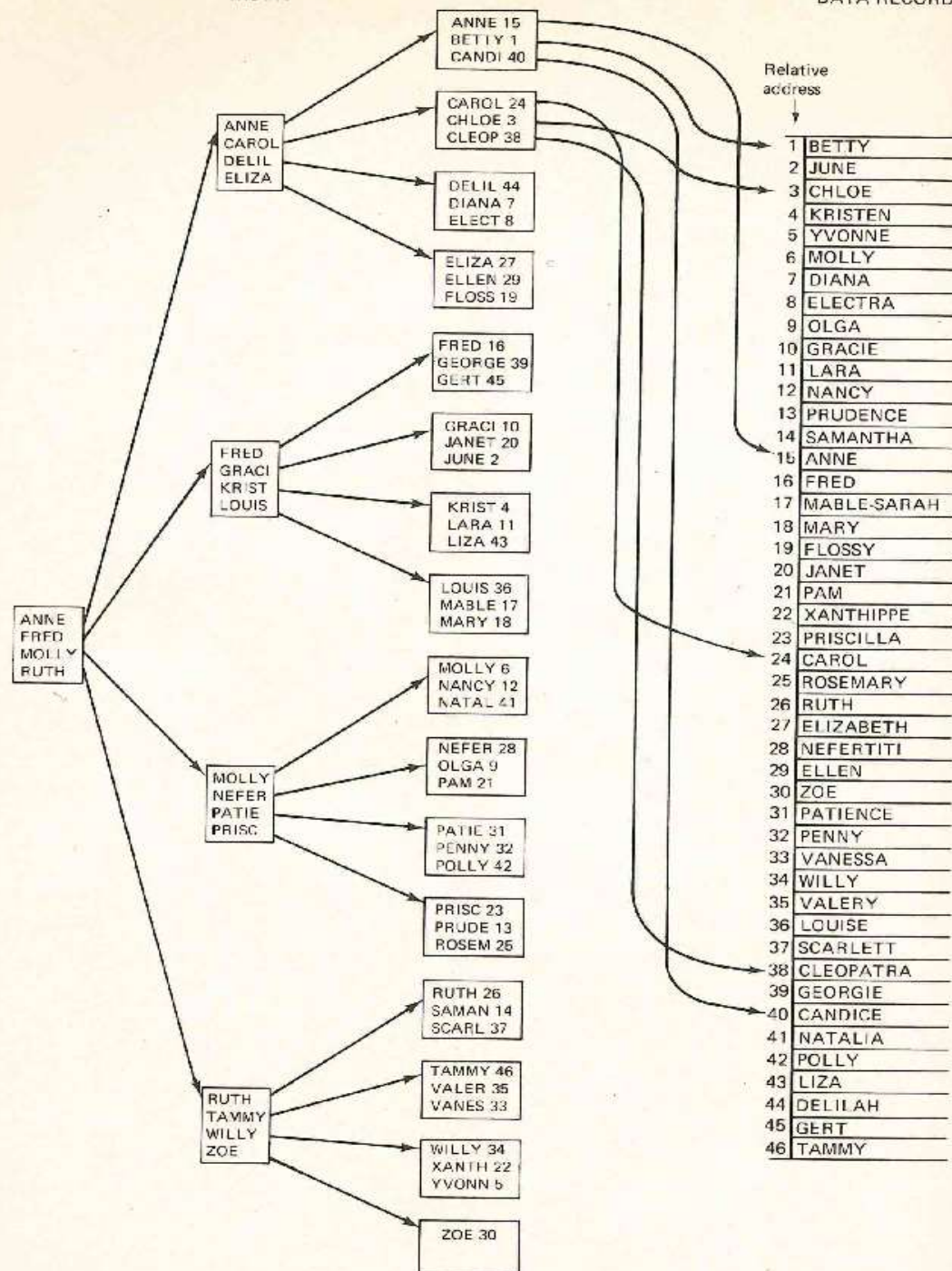
Indexes are easy to create!!!

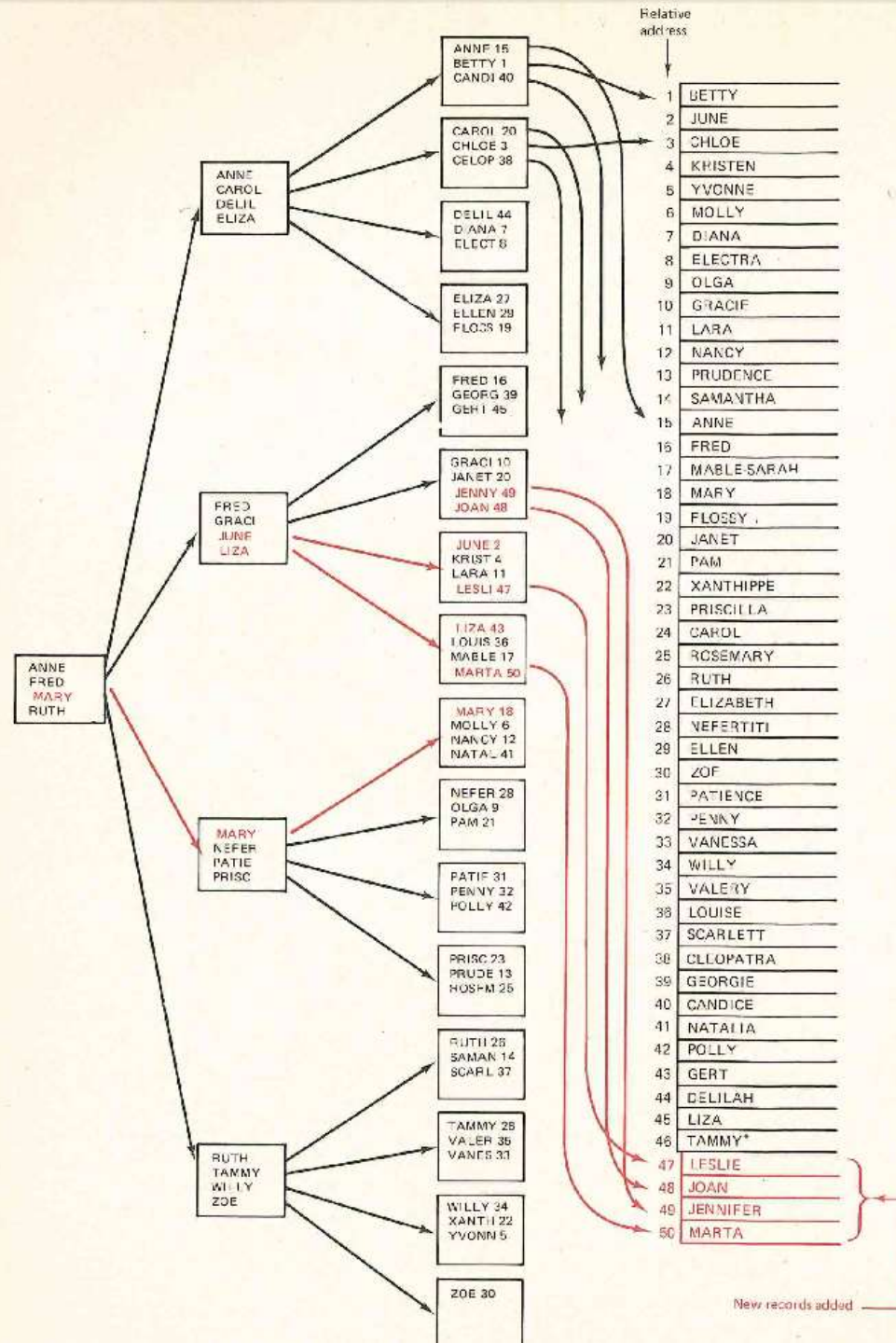
- Can add an index to a database with a simple SQL command:
 - Create index index_name on table (column_name);
- Understanding what happens when an index is created requires a basic understanding of indexing and file organization.

File organization and access concepts

- File organization.
 - The physical arrangement of data in a file into records and pages on secondary storage.
 - File organization dictates the physical placement of records on secondary storage.
- File access methods.
 - The steps involved in retrieving records.
 - File access methods dictate how data can be retrieved from secondary storage. Options include:
 - Sequential access from beginning. Sequential access from pre-defined point.
 - Backwards from end. Backwards from pre-defined point.
 - Direct. (not really direct – has to go through a series of indices or through a hashing algorithm)







What is an index?

- An additional physical file (just like a table).
- An index is a sorted list of logical pointers stored along with the actual data.
- Benefit: Indexes provide faster direct data access.
- Drawbacks:
 - Indexes create slower data updates.
 - Indexes require periodic reorganization.

Types of indexes

- Clustered index (primary index – does not have to be the primary key)
- Non-clustered index (secondary indices)

Rules of thumb for applying indexes

- Use on larger tables.
- Use when a relatively small percentage of the table will be accessed.
- Index the primary key of each table.
- Index frequently used search attributes.
- Index attributes in SQL “ORDER BY” and “GROUP BY” commands.
- Use indexes heavily for non-volatile databases; limit the use of indexes for volatile databases.
- Avoid indexing attributes that consist of long character strings.

Issues in indexing

- Indexes affect table maintenance performance.
 - Each time an add or delete is performed, the index must be updated along with the data.
 - Depending on the size of the database, these index updates can be extremely time-consuming.
 - Imagine the problems with having an index declared for every attribute.
- Solutions:
 - Remove indexes prior to batch updates.
 - Recreate indexes after the batch update is finished.
 - Consider using a batch procedure to create indexes after a table has been updated, and before queries are run.

Pop Quiz!

The purpose of creating an index on a table is:

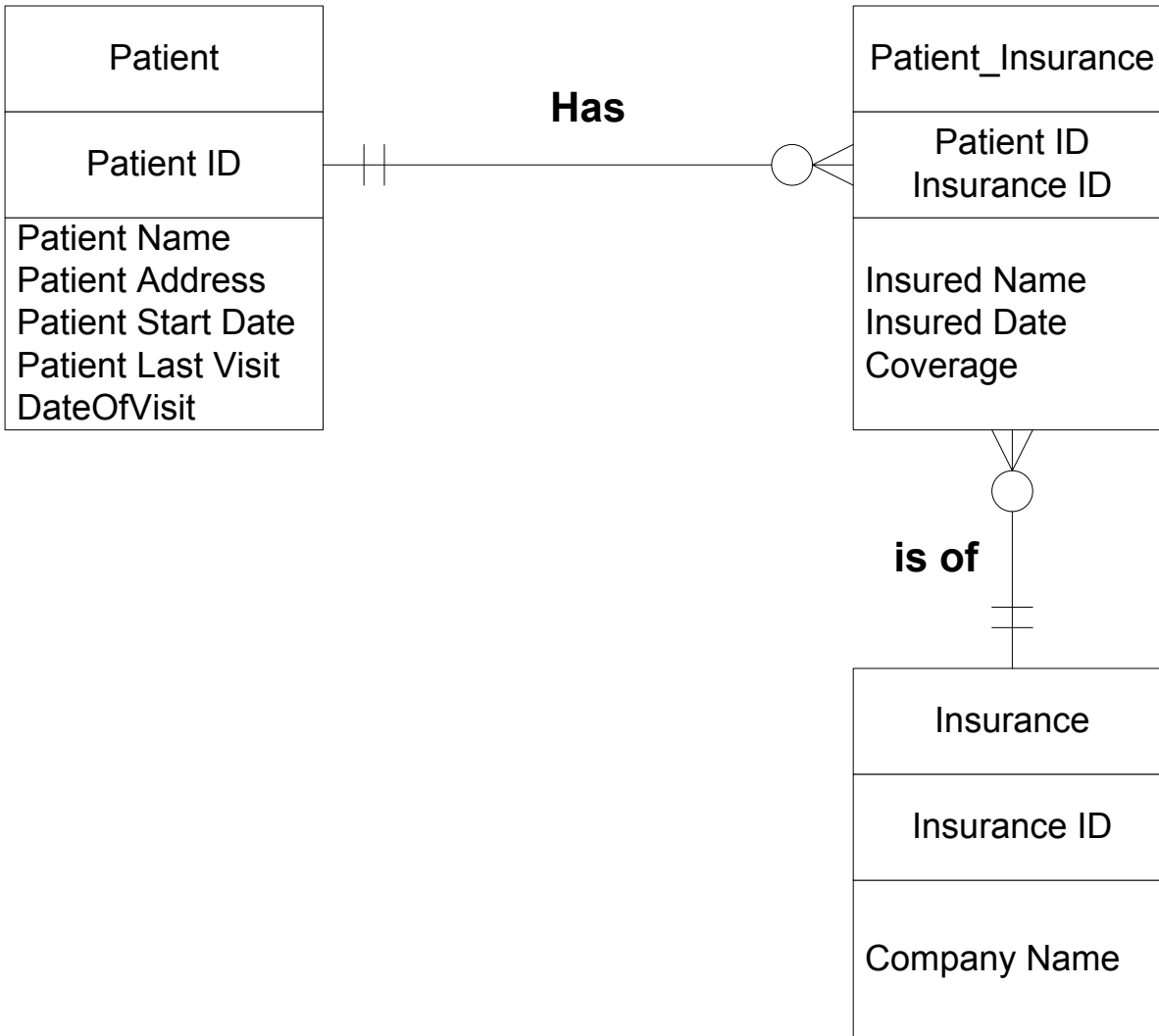
- a) To enhance the hardware, especially memory, of a database system.
- b) To simplify access to data.
- c) To process batched transactions more quickly.
- d) To provide faster direct access to data.

Improving performance with denormalization

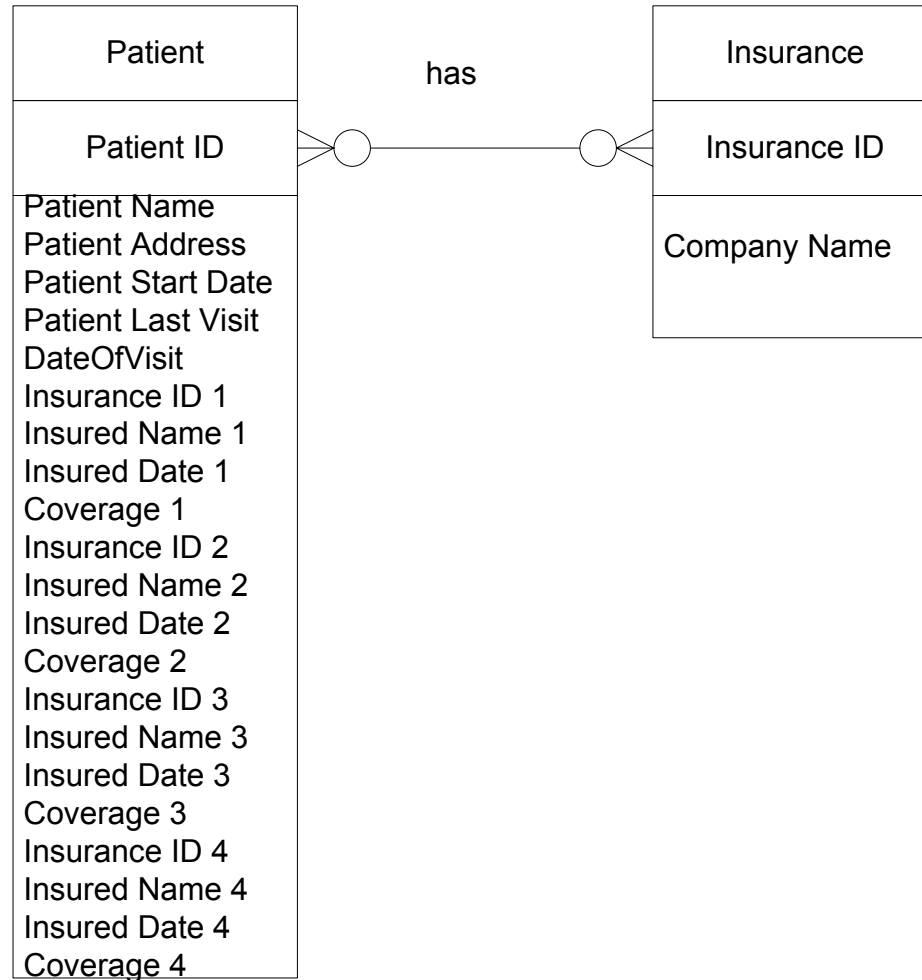
- Modify the degree of normalization.
 - Recognize that joins require much time when used in queries.
 - More joins = more time.
 - Combine entities with 1:1 relationship into a single entity.
 - Combine entities with 1:m relationship into a single entity. Usually done with brief repeating groups.

Example for denormalization

- Example:
 - A patient can have up to 4 insurance companies.
 - Patient is a strong entity. Insurance company is a strong entity.
 - Normally, the repeating group of insurance companies would be in a separate intersection entity relating a patient to one or more insurance companies.
- Diagram on next page



Insurance example - Denormalized



Patient
Patient ID
Patient Name Patient Address Patient Start Date Patient Last Visit Date Insurance ID 1 CompanyName1 Insured Name 1 Insured Date 1 Coverage 1 Insurance ID 2 CompanyName2 Insured Name 2 Insured Date 2 Coverage 2 Insurance ID 3 CompanyName3 Insured Name 3 Insured Date 3 Coverage 3 Insurance ID 4 CompanyName4 Insured Name 4 Insured Date 4 Coverage 4

Or Totally
De-Normalize it
back to a
spreadsheet in zero
normal form...

Issues in denormalization

- Can be risky.
 - Introduces potential for data redundancy.
 - Can result in data anomalies.
- Should be documented.
 - This documentation must be maintained as an “audit path” to the actual implementation of the database.
 - Logical data model details fully normalized database with an ERD.
 - Physical data model will show denormalized database with an ERD.
 - Include in the documentation the reasons for denormalization.

Improving performance with partitioning

- Create smaller or fewer tables.
- Sub-divide the database based on pre-defined factors.
 - **Horizontal partitioning:** split rows into separate tables. Could do it based on geographical region, date, or data types.
 - **Vertical partitioning:** split columns into separate tables. Could do it based on application types (accounting, manufacturing, or inventory control).

Improving performance with derived data

- Derived or calculated data is usually not included in a database.
 - Not ever included on a logical data model.
 - Examples of derived data include: extended price, total amount, total pay, etc.
- Problems with including derived data in a database:
 - What happens when the underlying data is changed? How do you ensure that the derived data will also be changed?
 - For example, let's say that the total of an order is kept in the database. What happens when an item quantity changes, or an item price changes? The order total, if stored, must also be changed to reflect those changes in the underlying data.

When to include derived data

- Sometimes it is a good idea to include derived data in the physical database design:
 - Use when aggregate values are regularly retrieved.
 - Use when aggregate values are costly to calculate.
 - Permit updating only of source data.
 - Do not put derived rows in same table as table containing source data.
- Examples of derived data frequently stored in databases:
 - Student class standing.
 - Order and invoice total.
 - Credit card balance.
 - Checking account balance.

Pop Quiz!

Which of the following is an example of a derived attribute?

- a. EmployeeID.
- b. Customer name.
- c. The price of an item purchased.
- d. The quantity of an item purchased.
- e. A checking account balance.

Which of the following is good advice about improving the performance of a very large, **volatile** database used for transaction processing?

- a. Include many indexes on fields in the database.
- b. Purchase more standard access magnetic disk.
- c. Purchase a faster processor.
- d. Denormalize the database.
- e. Purchase an in-memory database management system.

Organizations must manage data resources

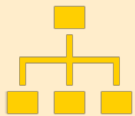


Examples of data stored by an organization

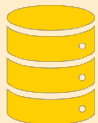
Current transaction data.

Historical data for decision making.

External data of interest for decision making.



All must be designed, implemented and maintained.



All data does not have to be stored together.

Can improve efficiency by understanding what data is best stored in separate databases.

IS475/675 Agenda: Week 15

Monday

- Review answer to HW#9.
- Describe HW#10.
- Finish discussing how a DBMS supports partitioned applications.
- Review/refresh logical vs. physical database design.
- Begin discussing how a DBMS can optimize the efficiency of a database.

Wednesday

- Discuss how a DBMS and a database designer can optimize the efficiency of a database.
- Identify the different databases in an organization
- Present how data is moved among databases.
- Introduce organizational data architecture.

Ways to Improve performance

1. By optimizing use of existing resources.
2. By using better or more resources.
3. By creating indexes.
4. By denormalizing the database.
5. By partitioning the database.
6. By storing derived data.
7. By creating procedures to differentiate and manage data.

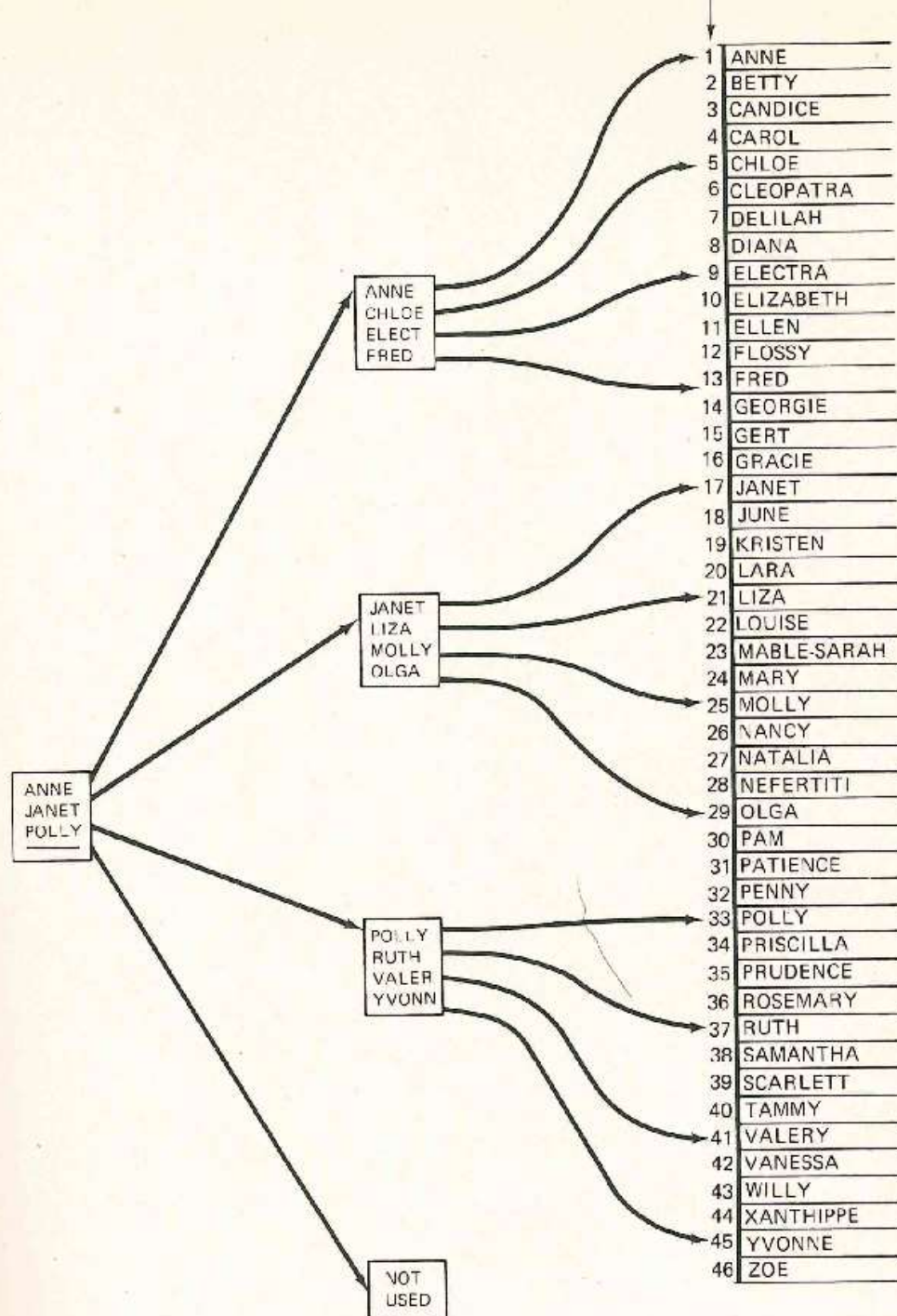
Indexes are the single most important tool a database **programmer and/or administrator** can use to improve the performance of a database.

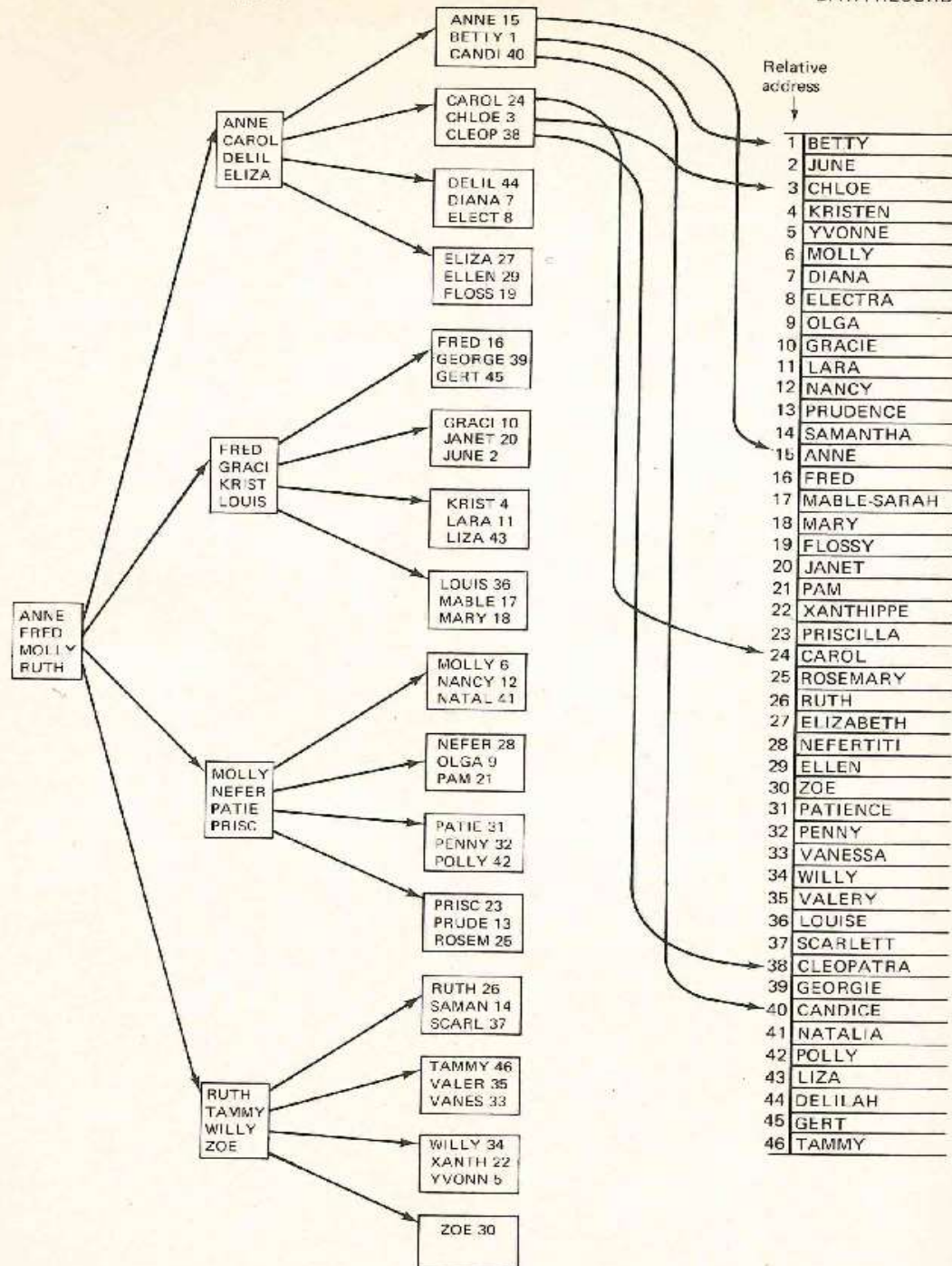
Indexes are easy to create!!!

- Can add an index to a database with a simple SQL command:
 - Create index index_name on table (column_name);
- Understanding what happens when an index is created requires a basic understanding of indexing and file organization.

File organization and access concepts

- File organization.
 - The physical arrangement of data in a file into records and pages on secondary storage.
 - File organization dictates the physical placement of records on secondary storage.
- File access methods.
 - The steps involved in retrieving records.
 - File access methods dictate how data can be retrieved from secondary storage. Options include:
 - Sequential access from beginning. Sequential access from pre-defined point.
 - Backwards from end. Backwards from pre-defined point.
 - Direct. (not really direct – has to go through a series of indices or through a hashing algorithm)



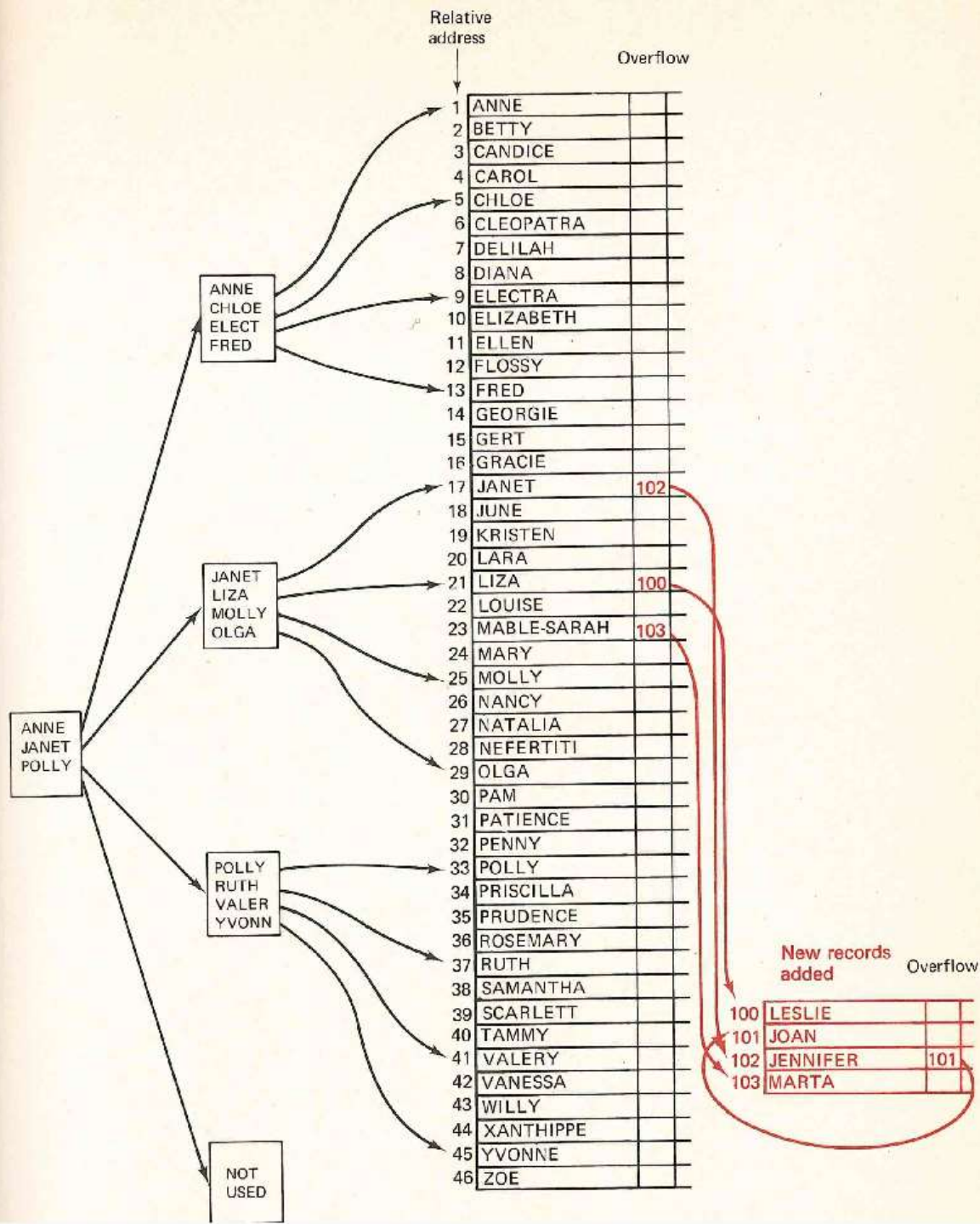


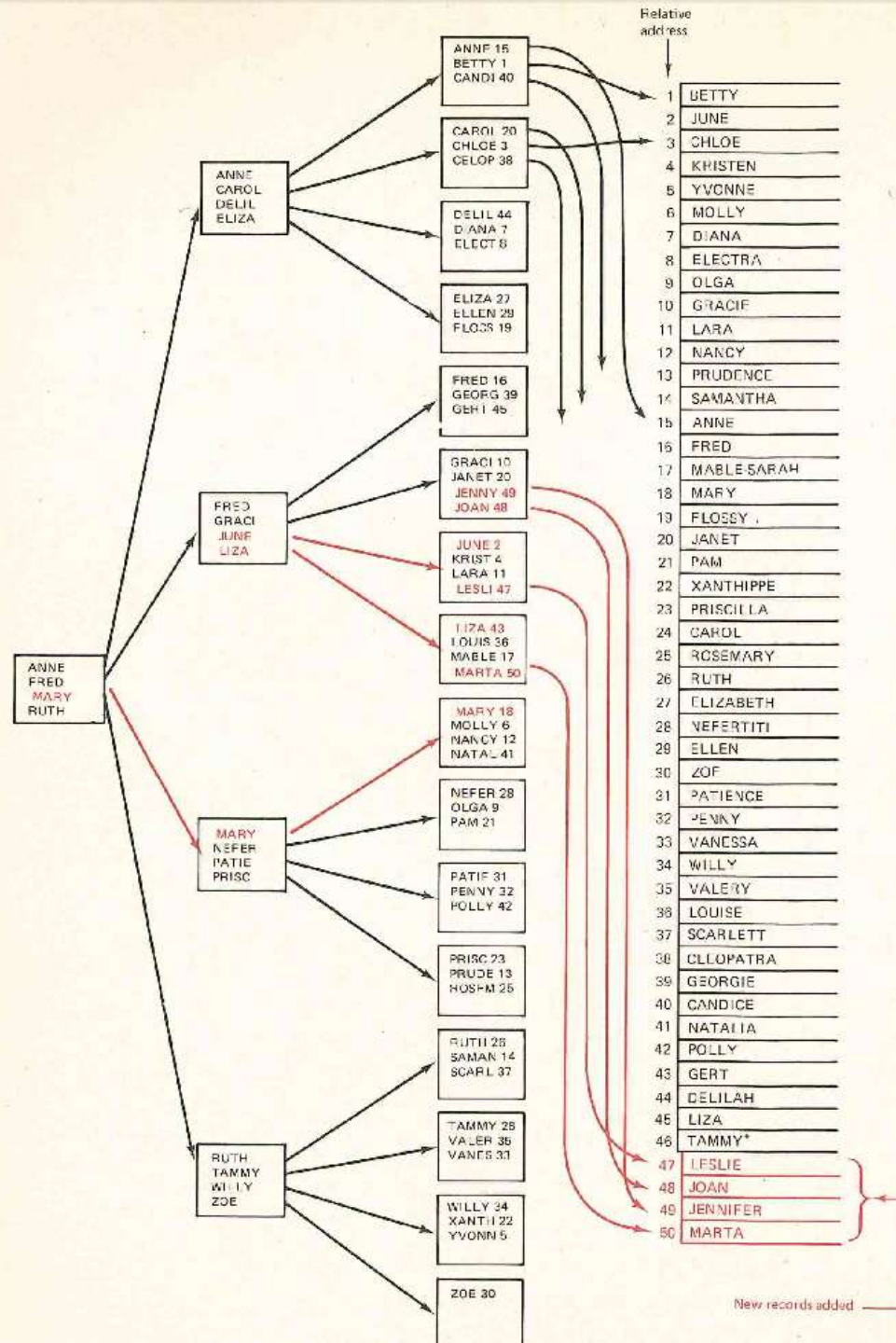
What is an index?

- An additional physical file (just like a table).
- An index is a sorted list of logical pointers stored along with the actual data.
- Benefit: Indexes provide faster direct data access.
- Drawbacks:
 - Indexes create slower data updates.
 - Indexes require periodic reorganization.

Types of indexes

- Clustered index (primary index – does not have to be the primary key)
- Non-clustered index (secondary indices)





Rules of thumb for applying indexes

- Use on larger tables.
- Use when a relatively small percentage of the table will be accessed.
- Index the primary key of each table.
- Index frequently used search attributes.
- Index attributes in SQL “ORDER BY” and “GROUP BY” commands.
- Use indexes heavily for non-volatile databases; limit the use of indexes for volatile databases.
- Avoid indexing attributes that consist of long character strings.

Issues in indexing

- Indexes affect table maintenance performance.
 - Each time an add or delete is performed, the index must be updated along with the data.
 - Depending on the size of the database, these index updates can be extremely time-consuming.
 - Imagine the problems with having an index declared for every attribute.
- Solutions:
 - Remove indexes prior to batch updates.
 - Recreate indexes after the batch update is finished.
 - Consider using a batch procedure to create indexes after a table has been updated, and before queries are run.

Pop Quiz!

The purpose of creating an index on a table is:

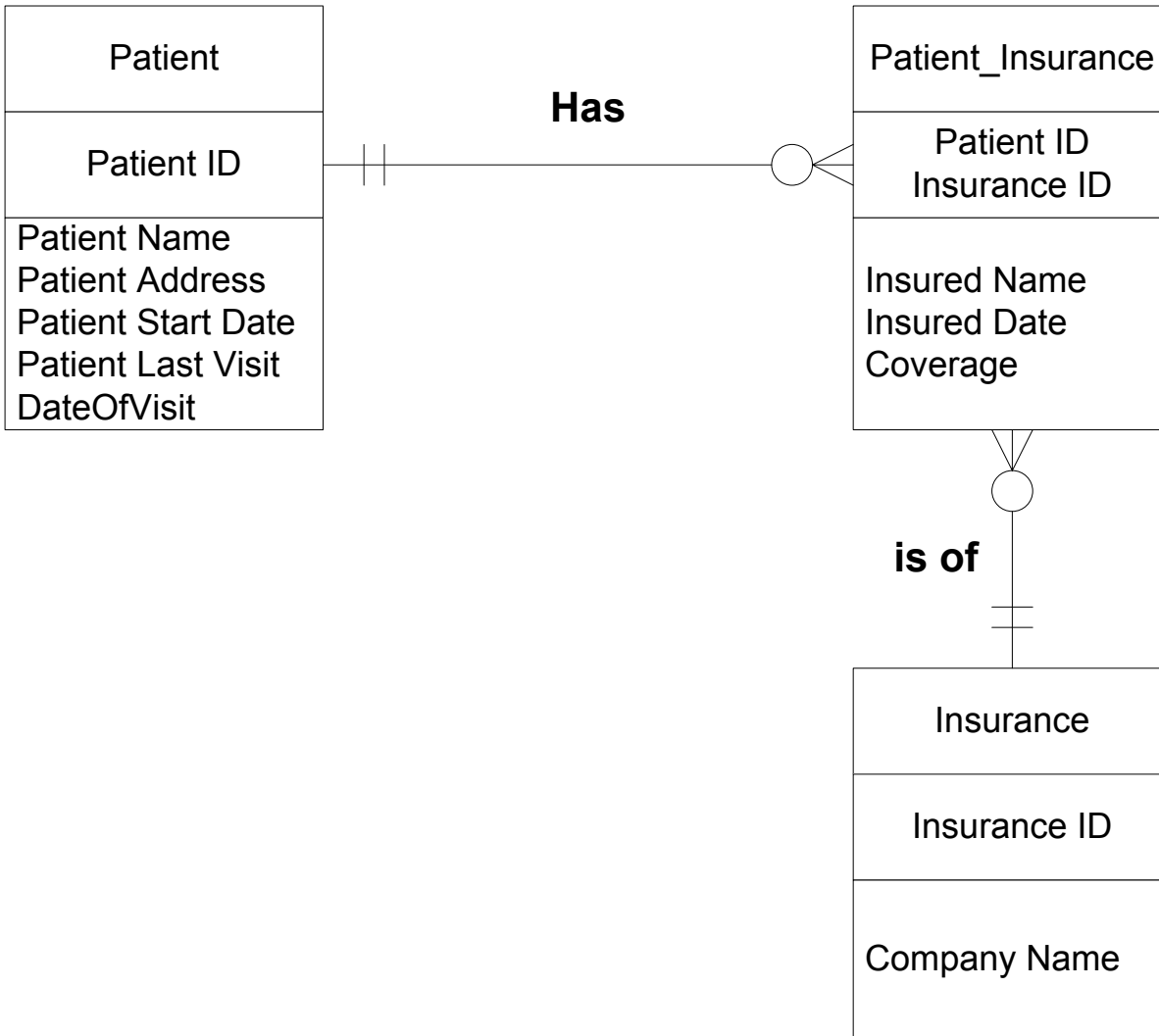
- a) To enhance the hardware, especially memory, of a database system.
- b) To simplify access to data.
- c) To process batched transactions more quickly.
- d) To provide faster direct access to data.

Improving performance with denormalization

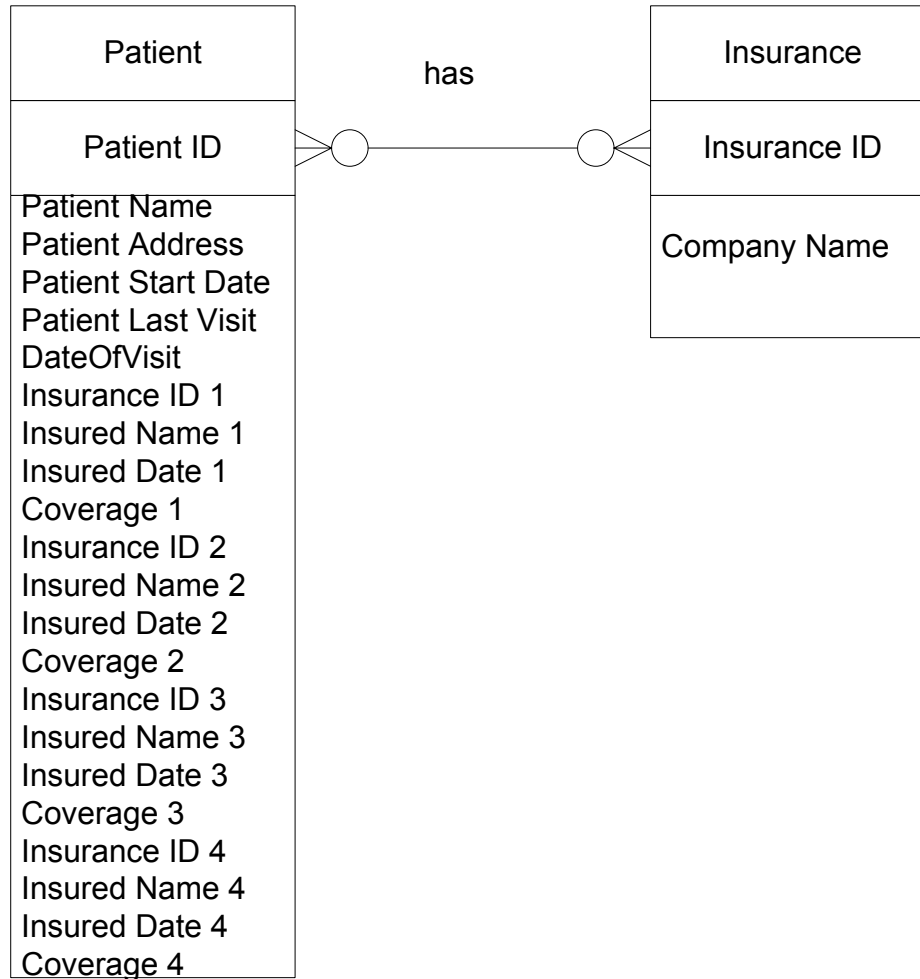
- Modify the degree of normalization.
 - Recognize that joins require much time when used in queries.
 - More joins = more time.
 - Combine entities with 1:1 relationship into a single entity.
 - Combine entities with 1:m relationship into a single entity. Usually done with brief repeating groups.

Example for denormalization

- Example:
 - A patient can have up to 4 insurance companies.
 - Patient is a strong entity. Insurance company is a strong entity.
 - Normally, the repeating group of insurance companies would be in a separate intersection entity relating a patient to one or more insurance companies.
- Diagram on next page



Insurance example - Denormalized



Patient
Patient ID
Patient Name Patient Address Patient Start Date Patient Last Visit Date Insurance ID 1 CompanyName1 Insured Name 1 Insured Date 1 Coverage 1 Insurance ID 2 CompanyName2 Insured Name 2 Insured Date 2 Coverage 2 Insurance ID 3 CompanyName3 Insured Name 3 Insured Date 3 Coverage 3 Insurance ID 4 CompanyName4 Insured Name 4 Insured Date 4 Coverage 4

Or Totally
De-Normalize it
back to a
spreadsheet in zero
normal form...

Issues in denormalization

- Can be risky.
 - Introduces potential for data redundancy.
 - Can result in data anomalies.
- Should be documented.
 - This documentation must be maintained as an “audit path” to the actual implementation of the database.
 - Logical data model details fully normalized database with an ERD.
 - Physical data model will show denormalized database with an ERD.
 - Include in the documentation the reasons for denormalization.

Improving performance with partitioning

- Create more, smaller tables with the intent that only some will be accessed depending on the application.
- Sub-divide the database based on pre-defined factors.
 - **Horizontal partitioning:** split rows into separate tables. Could do it based on geographical region, date, or data types.
 - **Vertical partitioning:** split columns into separate tables. Could do it based on application types (accounting, manufacturing, or inventory control).

Improving performance with derived data

- Derived or calculated data is usually not included in a database.
 - Not included on a logical data model.
 - Examples of derived data include: extended price, total amount, total pay, etc.
- Problems with including derived data in a database:
 - What happens when the underlying data is changed? How do you ensure that the derived data will also be changed?
 - For example, let's say that the total of an order is kept in the database. What happens when an item quantity changes, or an item price changes? The order total, if stored, must also be changed to reflect those changes in the underlying data.

When to include derived data

- Sometimes it is a good idea to include derived data in the physical database design:
 - Use when aggregate values are regularly retrieved.
 - Use when aggregate values are costly to calculate.
 - Permit updating only of source data.
 - Do not put derived rows in same table as table containing source data.
- Examples of derived data frequently stored in databases:
 - Student class standing.
 - Order and invoice total.
 - Credit card balance.
 - Checking account balance.

Pop Quiz!

Which of the following is an example of a derived attribute?

- a. EmployeeID.
- b. Customer name.
- c. The price of an item purchased.
- d. The quantity of an item purchased.
- e. A checking account balance.

Which of the following is good advice about improving the performance of a very large, **volatile** database used for transaction processing?

- a. Include many indexes on fields in the database.
- b. Purchase more standard access magnetic disk.
- c. Purchase a faster processor.
- d. Denormalize the database.
- e. Purchase a good DBMS that balances disk and memory utilization.

Organizations must manage data resources



Examples of data stored by an organization

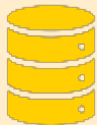
Current transaction data.

Historical data for decision making.

External data of interest for decision making.



All must be designed, implemented and maintained.



All data does not have to be stored together.

Can improve efficiency by understanding what data is best stored in separate databases.

What are the key objectives of data management?

■ **Protect and Control**

- Improve data quality.
- Protect and safeguard data.
- Assign ownership.
- Control change.

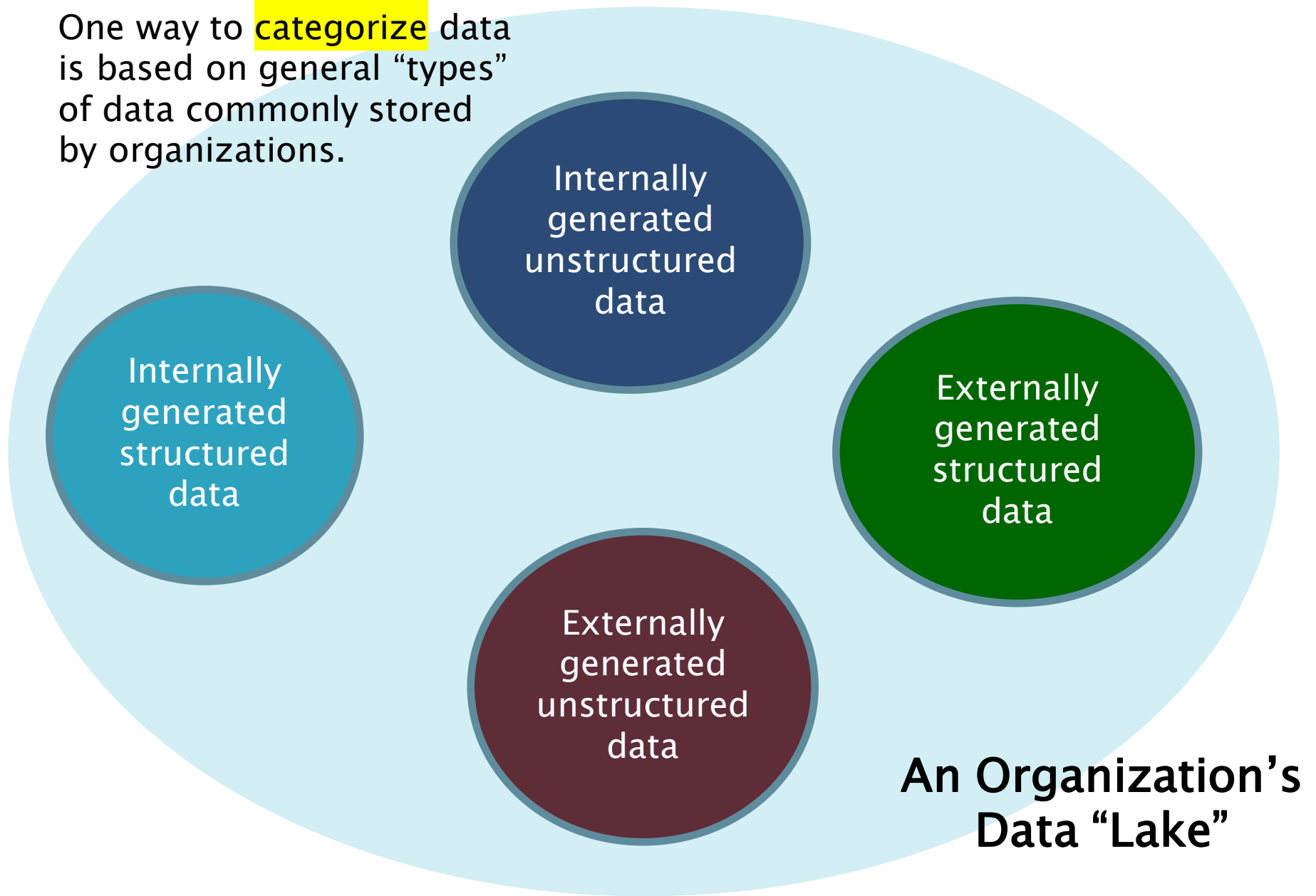
■ **Make Useful**

- Define data across the organization.
- Integrate data from a variety of different sources.
- Ensure data availability and accessibility.

■ **Adapt and Change**

- Encourage data use.
- Facilitate ongoing data evolution and acquisition.

One way to **categorize** data is based on general “types” of data commonly stored by organizations.



Structured vs. unstructured data

Structured Data

- Structure of data is pre-defined. Referred to as “schema on write.”
- Predictable data types such as numbers, text characters (also referred to as “strings”), dates.
- Examples for a company like Amazon: order data, return data, customer data, employee data, location data, product data, supplier data.
- Most often needs to be shared among people in an organization.
- Accuracy is critical to the financial reporting of the organization.
- Easy to search.

Unstructured Data

- Structure of data may or may not be pre-defined. If it is pre-defined, then it is “schema on write,” if it isn’t then it is “schema on read.”
- May not have a predictable data type.
- Examples for a company like Amazon: pictures, videos, sound recording, multi-media. This PowerPoint is unstructured data.
- May or may not need to be shared by people.
- Accuracy may not be critical to financial reporting.
- More difficult to search.

Internal vs. external data

Internal Data

- Internal organizational responsibility for data accuracy.
- May be input by employees.
- May be input by external stakeholders such as customers, suppliers, or clients.
- Is not usually perceived as an “asset” by the organization, even though it is used to support operations and decision making.
- The creation and maintenance of the data is written off as an expense.
- Examples: customer orders, purchase orders, employee time sheets, customer returns, email, PowerPoints, Word documents.

External data

- Externally generated and maintained.
- Organization using the data may not know the accuracy or veracity of the data.
- May be purchased by the organization to supplement data that is produced internally.
- May be considered an asset because it has a dollar value associated with it if purchased.
- Examples: census data, hospital quality rankings, school district testing results, zip code tables, Yelp reviews (for any company except Yelp), blog postings.

Classify the data!

Type of Data	Internally or externally generated?	Structured or unstructured?
Accounts payable data		
Organizational email		
Regional demographics (government census)		
Yelp reviews		
Internal organizational memos		
Amazon orders entered by customers over the web (from the perspective of Amazon)		

Data stored in an organization's
shared relational databases



Internally generated
structured data

Internally
generated
unstructured
data

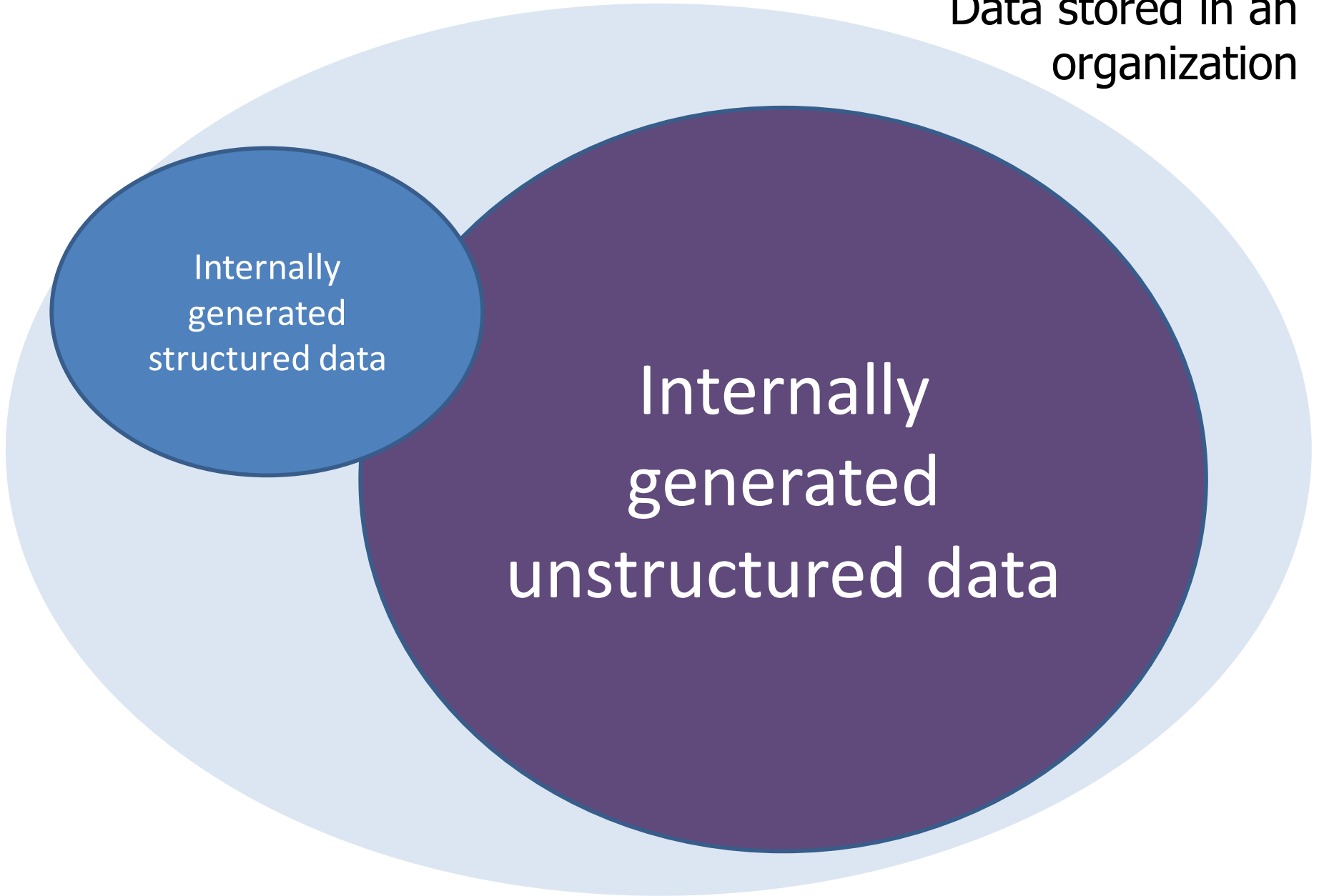
Externally
generated
unstructured
data

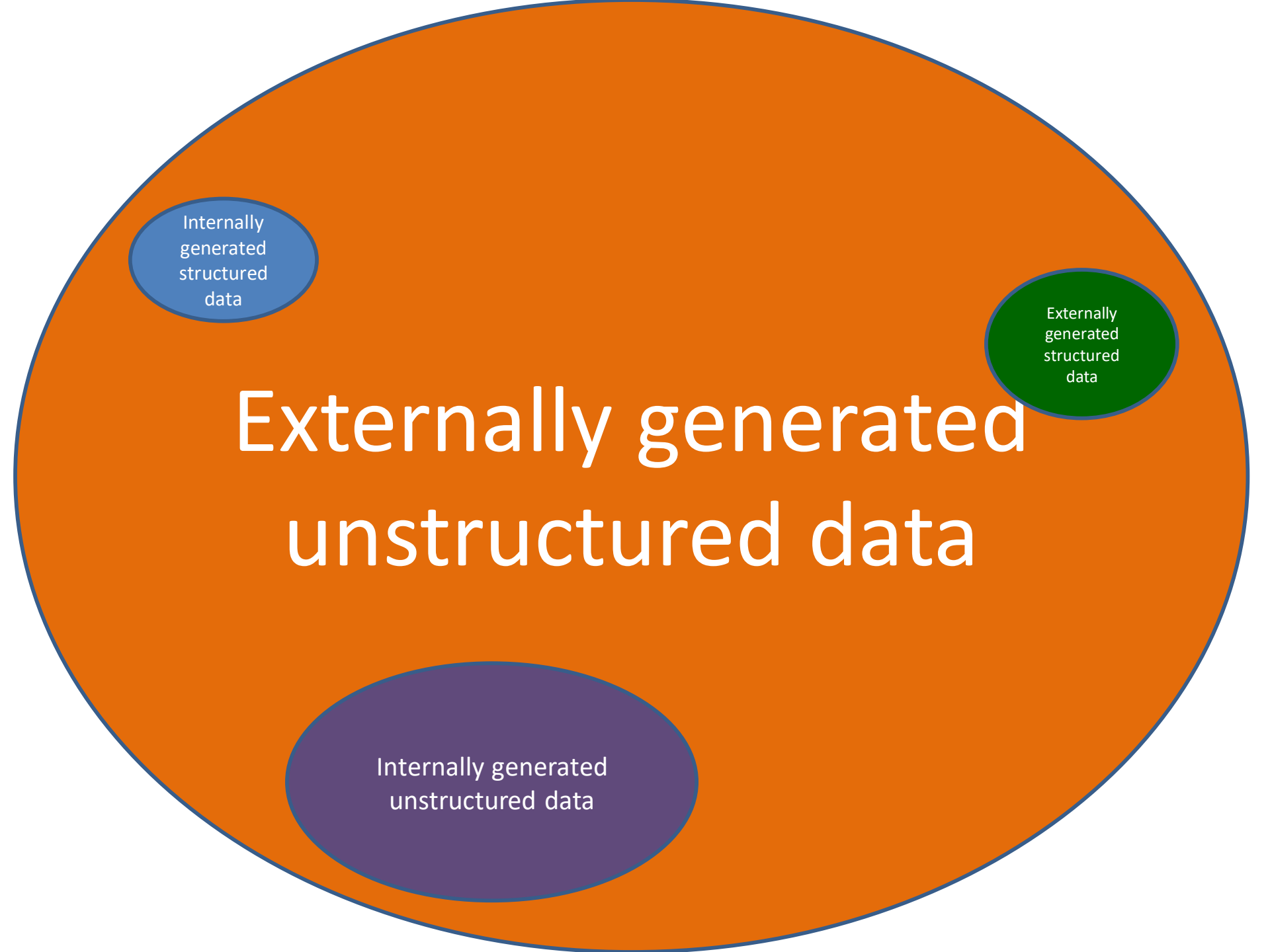
Externally
generated
structured
data

Data stored in an
organization

Internally
generated
structured data

Internally
generated
unstructured data





Internally
generated
structured
data

Externally
generated
structured
data

Externally generated unstructured data

Internally generated
unstructured data

“Managing” each type of data

- Internally generated structured data:

- Relational DBMS
- Schema on write
- Master data management
- Data stewards

- Internally generated unstructured data:

- Individual software solutions (i.e. Outlook Exchange Server; SharePoint)
- Relational DBMS
- Human data governance for sharing and retaining data

- Externally generated structured data:

- Relational DBMS.
- Schema on write.
- Purchased or obtained for a pre-defined purpose.
- Might cost money – might be an asset

- Externally generated unstructured data:

- Frequently referred to as “big data.”
- NoSQL type of database or Hadoop or relational?
- Schema on read.
- Management unclear at this time.

The “V’s” of Big Data

- **Volume**: The quantity of data collected. Volume alone does not define big data – database management systems have been handling large volumes of data for years.
- **Velocity**: The frequency of change. Big data arrives at high speed and from multiple sources.
- **Variety**: Different forms and sources of data. Both structured and unstructured data compose big data.
- **Veracity**: With big data, there is uncertainty of the accuracy of data. It isn’t always clear who entered the data and whether it can be verified as accurate.
- **Value**: Storing data is just storing data if it doesn’t provide value toward a meaningful goal. However, with big data the goal may not always be clear at first glance and may not have value until people have the data and decide how it can be used.

What type of data in an organization's "data lake" composes the largest category of data currently stored in an organization's **relational** database?

- A. Internally generated structured data
- B. Internally generated unstructured data.
- C. Externally generated structured data.
- D. Externally generated unstructured data.

What type of data in an organization's "data lake" composes the largest amount of data currently collected and digitally stored by most organizations?

- A. Internally generated structured data
- B. Internally generated unstructured data.
- C. Externally generated structured data.
- D. Externally generated unstructured data.

“Information Gap also called the Information Paradox”

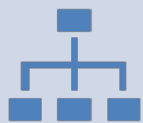
Organizations are drowning in data, but people in those organizations are starving for information to help them solve problems.



Why do we have an information gap?

- **Fragmented development.** Today is not our first rodeo. Information systems and their supporting databases are built over many years frequently answering an urgent need for specific transaction processing.
 - Limited resources make many organizations focus on one-thing-at-a-time.
 - Inter-relationships among systems may not always be the major focus.
 - May buy differing hardware and software platforms to answer a specific need.
- **Accounting transaction focus.** Organizations focus on the transaction. They focus on the need for transactional reporting rather than managerial decision making.
- **Too much data.** The vast quantity of data gathered from various sources has made many IS departments so constrained for resources, that the focus is on storing data rather than transforming it into information.

Another way to **categorize** data is based on the type of system that generates or uses the data



Operational system: a system that is used to run a business. Based on individual transactions. Frequently called a “system of record”.



Informational system: A system designed to support decision making. Includes data stored over time so that it can be used as historical data for prediction analytic systems.

Operational Systems

- Synonyms: Transaction processing system, system of record.
- Operational systems:
 - Are frequently based on financial transactions.
 - Usually have a limited, pre-defined set of transactions.
 - Usually have a limited, pre-defined set of decisions that must be supported from the system.
 - Usually need fast answers to questions.
 - Serve as the source data for informational systems.

Informational Systems

- Synonyms: Business intelligence systems (BI), decision support systems (DSS), executive information systems (EIS).
- Informational systems:
 - Centralize data that are scattered throughout disparate operational systems.
 - Clean the data to prior to loading so that it is as accurate as possible for decision making.
 - Are separate from operational systems so that the complex queries required to gather the data for decision making do not impact the performance requirements of the operational system.
 - Support decisions/questions that can usually be answered slowly because they do not have a direct impact on operations.

We use data to answer management questions in both systems

Operational Questions

- What products are due to be received today?
- What products were received today?
- What is the price for ProductID 1224 on PO#0667?
- When is the next due date for ProductID 8992?
- Which employee received the products for PO#0667?
- Which employee placed the most purchase orders this month?

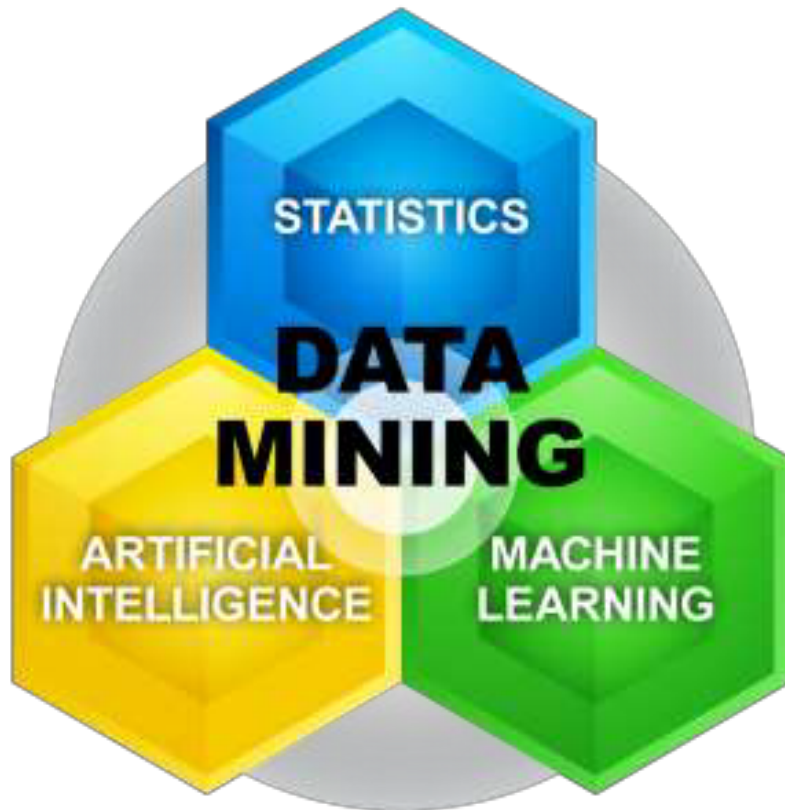
Informational Questions

- Which vendor gives us the best price for ProductID 8992?
- Which vendor delivers ProductID 8992 most reliably (on time and in best condition)?
- Which Product Type is increasing in price most significantly over the last six months?
- Which Product Type is decreasing in price most quickly?
- Do employees place purchase orders with the same vendors, or do they differentiate based on price or reliability?

People want to be able to answer informational questions that require long term data storage; these questions sometimes require very long-term data storage with very large datasets.

OLAP (online analytical processing)	Data Mining
Which products did we buy the most last year?	Which types of products will we need the most next year?
Which vendors provided the best price for our most-used products?	What are the characteristics of the vendors that delivered the most products on-time and of the best quality?
What were our highest selling products in the western U.S. last year?	What products will be highest selling in the western U.S. next year?
Which geographic location bought the most profitable products from us last year?	What are the general characteristics of our customers vs. the demographics of various regions across the world?

Data mining



- Data mining tools:
 - analyze the data;
 - uncover patterns hidden in the data;
 - form computer models based on the findings; and
 - use the models to predict business behavior.
- Proactive tools.
- Based on artificial intelligence software such as decision trees, neural networks, fuzzy logic systems, inductive nets and classification networking.

Create separate informational system database(s) from operational system database(s) because they are different based on these characteristics



Volume: The quantity of data collected is significantly larger than what is stored to handle the day-to-day transaction processing in an organization. May slow processing to store both “old” and “current” data in the same tables.



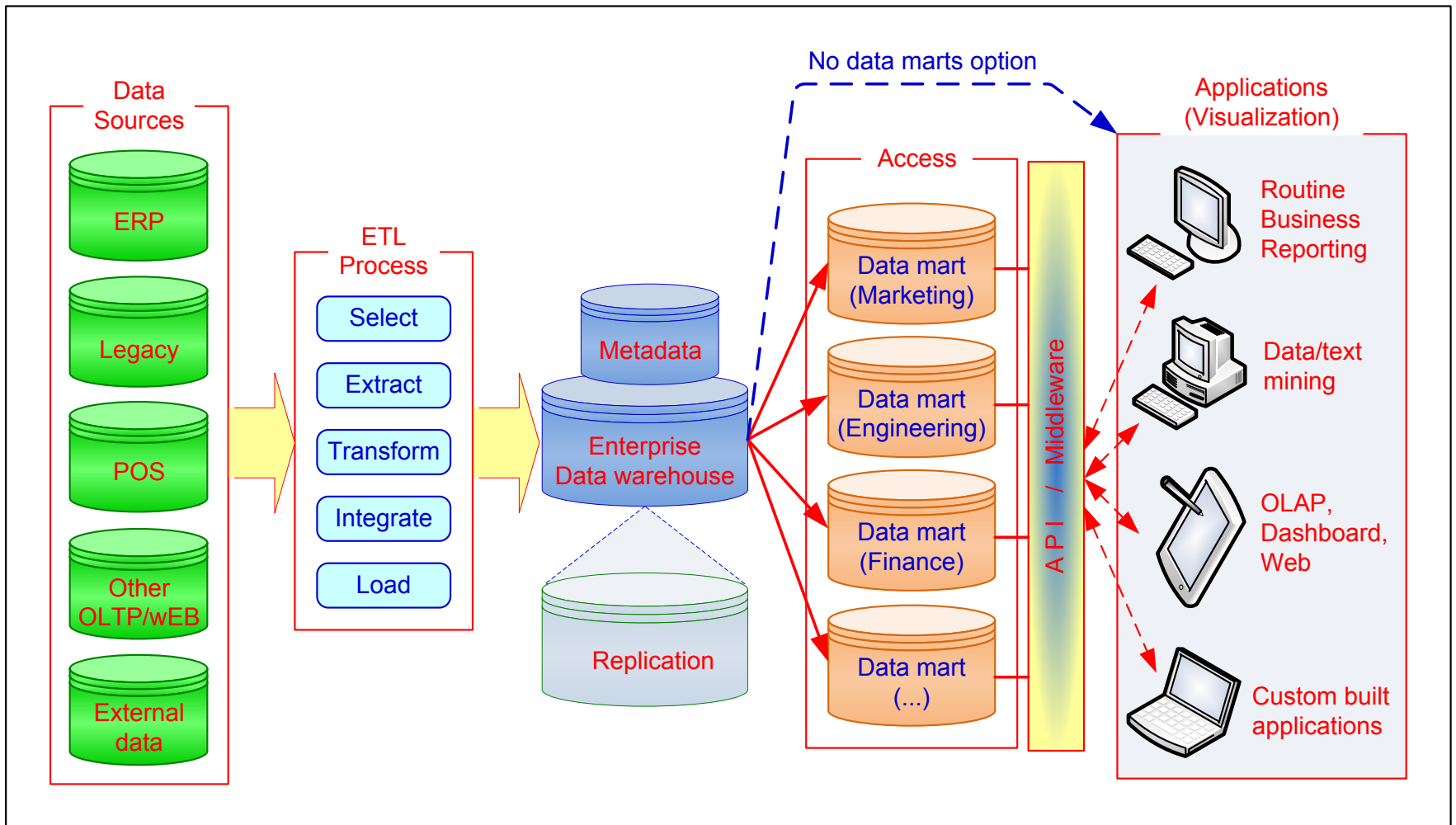
Time: Storing data over time.



Data Capture: Data will come from pre-existing digital data; it isn’t re-entered through human capture methods.



Use: Data will be used for more medium and long-term decision making.



What are these databases?

- **System of record database:** Represents the source data in an organization. Data is usually entered by people or automated data input (i.e. RFID, IOT, scanners).
- **Enterprise data warehouse:** A centralized, integrated database that is the single source of all data made available to end users for decision support applications (informational systems). Data is obtained from source data.
- **Data mart:** A data warehouse that is limited in scope whose data are obtained by selecting and summarizing data either from a data warehouse or from a separate ETL process from source data.

Integrated data management framework

	Operational Data	Informational Data	
		Data Warehouse/Data Mart	Big Data
Who generates?	Internal Mainly structured	Internal	External
What?	Transaction processing systems	Structured	Both structured and unstructured
How stored & accessed?	Relational technologies - DBMS	Relational technologies - DBMS	Both relational and non-relational technologies – may be DBMS or a “data lake”
How designed?	Logical data modeling using the relational data model	Logical data modeling (schema on write)	No advance modeling (schema on read)
Governance?	Strong policies and management in place to handle financial and operational transactions	Strong policies and management	Depends on the value of the data

A “system of record”:

- A. Is a term for the authoritative data source for a given data element.
- B. Contains the data that people rely on as accurate transaction data.
- C. Is the system where a data element is usually originally collected.
- D. All of the above.

Organizations may create different databases to support operational questions and informational questions because:

- a. Operational questions can be answered more slowly than informational questions.
- b. Operational questions do not usually require data stored at a very detailed level of granularity.
- c. Informational questions may require data integrated from a variety of different systems of record.
- d. Operational questions tend to be very complex and require significant processing time and data stored over time to answer fully.