4A - Systems, Databases, Networks

Bimal R. Desai, MD, MBI, FAAP, FAMIA
Children's Hospital of Philadelphia

Clinical Informatics
Board Review Course

Clinical Informatics Subspecialty Delineation of Practice (CIS DoP)

Domain 1: Fundamental Knowledge and Skills (no Tasks are associated with this Domain which is focused on fundamental knowledge and skills)

Clinical Informatics

K001. The discipline of informatics (e.g., definitions, history, careers, professional organizations)
K002. Fundamental informatics concepts, models, and theories
K003. Core clinical informatics literature (e.g., foundational literature, principle journals, critical analysis of literature, use of evidence to inform practice)
K004. Descriptive and inferential statistics
K005. Health Information Technology (HIT) principles and science
K006. Computer programming fundamentals and computational thinking
K007. Basic systems and network architectures
K008. Basic database structure, data retrieval and analytics techniques and tools
K009. Development and use of interoperability/exchange standards (e.g., Fast Health Interoperability Resources [FHIR], Digital Imaging and Communications in Medicine [DICOM])
K010. Development and use of transaction standards (e.g., American National Standards Institute X12)
K011. Development and use of messaging standards (e.g., Health Level Seven [HL7] v2)
K012. Development and use of ancillary data standards (e.g., imaging and Laboratory Information System[LIS])
K013. Development and use of data model standards
K014. Vocabularies, terminologies, and nomenclatures (e.g., Logical Observation Identifiers Names and Codes [LOINC], Systematized Nomenclature of Medicine --Clinical Terms [SNOMED-CT], RxNorm, International Classification Of Diseases[ICD], Current Procedural Terminology [CPT])
K015. Data taxonomies and ontologies
K016. Security, privacy, and confidentiality requirements and practices
K017. Legal and regulatory issues related to clinical data and information sharing
K018. Technical and non-technical approaches and barriers to interoperability
K019. Ethics and professionalism

The Health System

K020. Primary domains of health, organizational structures, cultures, and processes (e.g., health care delivery, public health, personal health, population health, education of health professionals, clinical research)
K021. Determinants of individual and population health
K022. Forces shaping health care delivery and considerations regarding health care access
K023. Health economics and financing
K024. Policy and regulatory frameworks related to the healthcare system
K025. The flow of data, information, and knowledge within the health system

Domain 2: Improving Care Delivery and Outcomes

K026. Decision science (e.g., Bayes theorem, decision analysis, probability theory, utility and preference assessment, test characteristics)
K027. Clinical decision support standards and processes for development, implementation, evaluation, and maintenance
K028. Five Rights of clinical decision support (i.e., information, person, intervention formats, channel, and point/time in workflow)
K029. Legal, regulatory, and ethical issues regarding clinical decision support
K030. Methods of workflow analysis
K031. Principles of workflow re-engineering
K032. Quality improvement principles and practices (e.g., Six Sigma, Lean, Plan-Do-Study-Act [PDSA] cycle, root cause analysis)
K033. User-centered design principles (e.g., iterative design process)
K034. Usability testing
K035. Definitions of measures (e.g., quality performance, regulatory, pay for performance, public health surveillance)
K036. Measure development and evaluation processes and criteria
K037. Key performance indicators (KPIs)
K038. Claims analytics and benchmarks
K039. Predictive analytic techniques, indications, and limitations
K040. Clinical and financial benchmarking sources (e.g., Gartner, Healthcare Information and Management Systems Society [HIMSS] Analytics, Centers for Medicare and Medicaid Services [CMS], Leapfrog)
K041. Quality standards and measures promulgated by quality organizations (e.g., National Quality Forum [NQF], Centers for Medicare and Medicaid Services [CMS], National Committee for Quality Assurance [NCQA])
K042. Facility accreditation quality and safety standards (e.g., The Joint Commission, Clinical Laboratory Improvement Amendments [CLIA])
K043. Clinical quality standards (e.g., Physician Quality Reporting System [PQRS], Agency for Healthcare Research and Quality [AHRQ], National Surgical Quality Improvement Program [NSQIP], Quality Reporting Document Architecture [QRDA], Health Quality Measure Format [HQMF], Council on Quality and Leadership [CQL], Fast Health Interoperability Resources [FHIR] Clinical Reasoning)
K044. Reporting requirements
K045. Methods to measure and report organizational performance
K046. Adoption metrics (e.g., Electronic Medical Records Adoption Model [EMRAM], Adoption Model for Analytics Maturity [AMAM])
K047. Social determinants of health
K048. Use of patient-generated data
K049. Prediction models
K050. Risk stratification and adjustment
K051. Concepts and tools for care coordination
K052. Care delivery and payment models

Domain 3: Enterprise Information Systems

K053. Health information technology landscape (e.g., innovation strategies, emerging technologies)
K054. Institutional governance of clinical information systems
K055. Information system maintenance requirements
K056. Information needs analysis and information system selection
K057. Information system implementation procedures
K058. Information system evaluation techniques and methods
K059. Information system and integration testing techniques and methodologies
K060. Enterprise architecture (databases, storage, application, interface engine)
K061. Methods of communication between various software components
K062. Network communications infrastructure and protocols between information systems (e.g., Transmission Control Protocol/Internet Protocol [TCP/IP], switches, routers)
K063. Types of settings (e.g., labs, ambulatory, radiology, home) where various systems are used
K064. Clinical system functional requirements
K065. Models and theories of human-computer (machine) interaction (HCI)
K066. HCI evaluation, usability engineering and testing, study design and methods
K067. HCI design standards and design principles
K068. Functionalities of clinical information systems (e.g., Electronic Health Records [EHR], Laboratory Information System [LIS], Picture Archiving and Communication System [PACS], Radiology Information System [RIS] vendor-neutral archive, pharmacy, revenue cycle)
K069. Consumer-facing health informatics applications (e.g., patient portals, mobile health apps and devices, disease management, patient education, behavior modification)
K070. User types and roles, institutional policy and access control
K071. Clinical communication channels and best practices for use (e.g., secure messaging, closed loop communication)
K072. Security threat assessment methods and mitigation strategies
K073. Security standards and safeguards
K074. Clinical impact of scheduled and unscheduled system downtimes
K075. Information system failure modes and downtime mitigation strategies (e.g., replicated data centers, log shipping)
K076. Approaches to knowledge repositories and their implementation and maintenance
K077. Data storage options and their implications
K078. Clinical registries
K079. Health information exchanges
K080. Patient matching strategies
K081. Master patient index
K082. Data reconciliation
K083. Regulated medical devices (e.g., pumps, telemetry monitors) that may be integrated into information systems
K084. Non-regulated medical devices (e.g., consumer devices)
K085. Telehealth workflows and resources (e.g., software, hardware, staff)

Domain 4: Data Governance and Data Analytics

K086. Stewardship of data
K087. Regulations, organizations, and best practice related to data access and sharing agreements, data use, privacy, security, and portability
K088. Metadata and data dictionaries
K089. Data life cycle
K090. Transactional and reporting/research databases
K091. Techniques for the storage of disparate data types
K092. Techniques to extract, transform, and load data
K093. Data associated with workflow processes and clinical context
K094. Data management and validation techniques
K095. Standards related to storage and retrieval from specialized and emerging data sources
K096. Types and uses of specialized and emerging data sources (e.g., imaging, bioinformatics, internet of things (IoT), patient-generated, social determinants)
K097. Issues related to integrating emerging data sources into business and clinical decision making
K098. Information architecture
K099. Query tools and techniques
K100. Flat files, relational and non-relational/NoSQL database structures, distributed file systems
K101. Definitions and appropriate use of descriptive, diagnostic, predictive, and prescriptive analytics
K102. Analytic tools and techniques (e.g., Boolean, Bayesian, statistical/mathematical modeling)
K103. Advanced modeling and algorithms
K104. Artificial intelligence
K105. Machine learning (e.g., neural networks, support vector machines, Bayesian network)
K106. Data visualization (e.g., graphical, geospatial, 3D modeling, dashboards, heat maps)
K107. Natural language processing
K108. Precision medicine (customized treatment plans based on patient-specific data)
K109. Knowledge management and archiving science
K110. Methods for knowledge persistence and sharing
K111. Methods and standards for data sharing across systems (e.g., health information exchanges, public health reporting)

Domain 5: Leadership and Professionalism

K112. Environmental scanning and assessment methods and techniques
K113. Consensus building, collaboration, and conflict management
K114. Business plan development for informatics projects and activities (e.g., return on investment, business case analysis, pro forma projections)
K115. Basic revenue cycle
K116. Basic managerial/cost accounting principles and concepts
K117. Capital and operating budgeting
K118. Strategy formulation and evaluation
K119. Approaches to establishing Health Information Technology (HIT) mission and objectives
K120. Communication strategies, including one-on-one, presentation to groups, and asynchronous communication
K121. Effective communication programs to support and sustain systems implementation
K122. Writing effectively for various audiences and goals
K123. Negotiation strategies, methods, and techniques
K124. Conflict management strategies, methods, and techniques
K125. Change management principles, models, and methods
K126. Assessment of organizational culture and behavior change theories
K127. Theory and methods for promoting the adoption and effective use of clinical information systems
K128. Motivational strategies, methods, and techniques
K129. Basic principles and practices of project management
K130. Project management tools and techniques
K131. Leadership principles, models, and methods
K132. Intergenerational communication techniques
K133. Coaching, mentoring, championing and cheerleading methods
K134. Adult learning theories, methods, and techniques
K135. Teaching modalities for individuals and groups
K136. Methods to assess the effectiveness of training and competency development
K137. Principles, models, and methods for building and managing effective interdisciplinary teams
K138. Team productivity and effectiveness (e.g., articulating team goals, defining rules of operation, clarifying individual roles, team management, identifying and addressing challenges)
K139. Group management processes (e.g., nominal group, consensus mapping, Delphi method)

Knowledge Statements from the DoP

K007. Basic systems and network architectures
K008. Basic database structure, data retrieval and analytics techniques and tools
K088. Metadata and data dictionaries
K089. Data life cycle
K090. Transactional and reporting/research databases
K091. Techniques for the storage of disparate data types
K092. Techniques to extract, transform, and load data
K098. Information architecture
K099. Query tools and techniques
K100. Flat files, relational and non-relational/NoSQL database structures, distributed file systems

Database

Any collection of related data (address book, spreadsheet, MS Access)

Database Management System (DBMS)

- Allows users to interact with DB and maintain structure, integrity
- Common features of DBMS
  - Define data types, structures, constraints
  - Construct data tables, store data on a storage medium
  - Manipulate data to create (insert), retrieve (read), update (edit), delete (sometimes abbreviated “CRUD”)
  - Share data via permissions, user access control; control concurrency
  - Protect against inappropriate access, hardware/software failure
  - Maintain & Optimize data structures

Flat File

Convenient, easy, ubiquitous

May require redundant data (eg: Louise Chen – have to remember to indicate in each cell that she is deceased)

Can't represent 1-to-many relationships easily (Louise has 2 meds)

Limited ability to enforce data integrity (multiple spellings of "yes" and "no")

Incomplete data represented as blank cells

My Patients & Medications			
Name	**DOB**	**Deceased?**	**Medication(s)**
John Smith	4/25/74	No	PROPRANOLOL
Jane Doe	8/12/58	N	Benadryl®
Jose Patel	2/19/88	y	amox
Louise Chen	12/8/69	YES	Acetaminophen, Topiramate

Relational Database

Defines association within and between **relations** (relation ~ table)

Each **attribute** (attribute ~ column) corresponds to a domain in the relation

Each **tuple** (tuple ~ row) describes an ordered list of elements, the order is important

Data **elements** (element ~ cell) have a data type that is consistent across that attribute. (VARCHAR, INT, DATE, LONG, etc)

Attributes can also have **constraints** (non NULL, auto-incrementing, cascading delete, Primary Key, Foreign Key) beyond the type constraint

Create and describe structure/constraints using "**Data Definition Language**" (DDL) which contains **metadata** (data about the data)

Further describe the data using a **Data Dictionary** (not just PK/FK, constraints, but also definitions of each field and its intended use)

Relational Database

The relation **schema** (table schema) is a description of the relation, its attributes, and the data types / rules associated with the relation.

A specific table that uses that schema is an **instance of that schema**

Adding new relations as easy as adding a new table, add an attribute by adding a column

In very simple terms these make it easy to know “everything that has one attribute”

Ex: “find all patients born in 1974”

Object-Relational Mapping (ORM)

Parallelism between OOP and RDBMS is very useful programmatically

- Object-oriented Class <-> DB Relation
- Instance <-> Tuple (a specific row) in a Relation (table) , where each row is a member of the class described by those attributes
- Attribute <-> Attribute (column), where the Value of that attribute is the element (content of the cell)
- Method (accessors, mutators) <-> database manipulation (CRUD) functions

Many modern programming languages use Object-Relational Mapping (ORM) either built-in or available as an extension

- Each class is mapped to a table
- Each attribute is mapped to a column
- Each method (getters/setters) mapped to a “CRUD” function
- Ex: Creating a new instance of a class automatically creates a row and populates data

Example Relational Schema

PATIENT		
Primary Key	**Pat_ID**	INT
	First Name	VARCHAR(50)
	Last Name	VARCHAR(50)
	DOB	DATE
	Is_Deceased	BOOLEAN

ORDER		
Primary Key	**Order_ID**	INT
	Med Name	VARCHAR(50)
Foreign Key to Pat_ID in PATIENT table	**Pat_ID**	INT

Example Relational Instance

PATIENT	Pat_ID	First_Name	Last_Name	DOB	Is_Deceased
	1001	John	Smith	4/25/74	N
	1002	Jane	Doe	8/12/58	N
	1003	Jose	Patel	2/19/88	N
	1004	Louise	Chen	12/8/69	Y

PK / FK relationship specifies how these tables are related

ORDER	Order_ID	Med_Name	Pat_ID
	991	amoxicillin	1003
	992	diphenhydramine	1002
	993	acetaminophen	1004
	994	topiramate	1004
	995	propranolol	1001

Pat_ID is the "Primary Key" in PATIENT
But is a "Foreign Key" in ORDER

Primary Key: an attribute that uniquely identifies a tuple (row)

Foreign Key: an attribute whose values must have matching values in the primary key of another table.

Structured Query Language

"SQL" – a family of related languages with different dialects specific to the RBDMS (MS-SQL, Oracle SQL, MySQL)

English-like with key words that allow for all functions common to DBMS ("CRUD" functions):

- **INSERT INTO** [table] **VALUES** [tuple]
- **SELECT** [columns] **FROM** [table] **WHERE** [constraints]
- **UPDATE** [table] **SET** [column] = [value] **WHERE** [condition]
- **DELETE FROM** [table] **WHERE** [column] = [value]

SQL Joins

Consider two tables with a PK/FK relationship

How do we write a **SELECT** statement that returns data from both tables?

PATIENT	Pat_ID	First_Name	Last_Name
	1001	John	Smith
	1002	Jane	Doe
	1003	Jose	Patel
	1004	Louise	Chen

ORDER	Order_ID	Med_Name	Pat_ID
	991	amoxicillin	1003
	992	diphenhydramine	1002
	993	acetaminophen	1004
	994	topiramate	1004
	995	propranolol	1001

SQL Joins

PATIENT	Pat_ID	First_Name	Last_Name
	1001	John	Smith
	1002	Jane	Doe
	1003	Jose	Patel
	1004	Louise	Chen

ORDER	Order_ID	Med_Name	Pat_ID
	991	amoxicillin	1003
	992	diphenhydramine	1002
	993	acetaminophen	1004
	994	topiramate	1004
	995	propranolol	1001

```
SELECT PATIENT.First_Name, PATIENT.Last_Name,
ORDER.Med_Name

FROM PATIENT

INNER JOIN ORDER

ON PATIENT.Pat_ID = ORDER.Pat_ID
```

SQL Joins

PATIENT	Pat_ID	First_Name	Last_Name
	1001	John	Smith
	1002	Jane	Doe
	1003	Jose	Patel
	1004	Louise	Chen

ORDER	Order_ID	Med_Name	Pat_ID
	991	amoxicillin	1003
	992	diphenhydramine	1002
	993	acetaminophen	1004
	994	topiramate	1004
	995	propranolol	1001

```
SELECT PATIENT.First_Name,
          PATIENT.Last_Name,
          ORDER.Med_Name
FROM PATIENT
INNER JOIN ORDER
ON PATIENT.Pat_ID = ORDER.Pat_ID
```

First_Name	Last_Name	Med_Name
John	Smith	propranolol
Jane	Doe	diphenhydramine
Jose	Patel	amoxicillin
Louise	Chen	acetaminophen
Louise	Chen	topiramate

SQL Joins

Note that this JOIN statement can also be written as a “WHERE” clause

These two statements have equivalent output

“OUTER” JOINS cannot be written using a WHERE clause

```
SELECT PATIENT.First_Name,
           PATIENT.Last_Name,
           ORDER.Med_Name
FROM PATIENT
INNER JOIN ORDER
ON PATIENT.Pat_ID = ORDER.Pat_ID
```

```
SELECT PATIENT.First_Name,
           PATIENT.Last_Name,
           ORDER.Med_Name
FROM PATIENT, ORDER
WHERE PATIENT.Pat_ID = ORDER.Pat_ID
```

NB: If there is no match in ORDER for a specific Pat_ID, that PATIENT will not appear in the resultset. That’s what is meant by the key word “INNER”

SQL Joins: Inner vs. Outer

How do you join tables where there is no one-to-one equivalence (not every row in A has a match in B, not every row in B has a match in A)?

	Faculty	
	User_Id	**User_name**
	54	Tom Payne
	30	Alexis Carter
	41	Bill Hersh
*	29	Bimal Desai
	12	Pesha Rubinstein

	Interests	
	User_ID	**Hobby**
	54	Hiking
	30	Languages
	41	Guitar
*	61	Home Improvement
	12	Reading

Inner Join
Only includes rows that match both tables

Faculty	
User_Id	**User_name**
54	Tom Payne
30	Alexis Carter
41	Bill Hersh
29	Bimal Desai
12	Pesha Rubinstein

Interests	
User_ID	**Hobby**
54	Hiking
30	Languages
41	Guitar
61	Home Improvement
12	Reading

```
SELECT Faculty.User_Name, Interests.Hobby

FROM Faculty

INNER JOIN Interests

ON Faculty.User_ID = Interests.User_ID
```

Faculty	Interests
Tom Payne	Hiking
Alexis Carter	Languages
Bill Hersh	Guitar
Pesha Rubinstein	Reading

Left Outer Join

Will include all rows in the left table, display blanks from right

Faculty	
User_Id	User_name
54	Tom Payne
30	Alexis Carter
41	Bill Hersh
29	Bimal Desai
12	Pesha Rubinstein

Interests	
User_ID	Hobby
54	Hiking
30	Languages
41	Guitar
61	Home Improvement
12	Reading

```
SELECT Faculty.User_Name, Interests.Hobby

FROM Faculty

LEFT OUTER JOIN Interests

ON Faculty.User_ID = Interests.User_ID
```

Faculty	Interests
Tom Payne	Hiking
Alexis Carter	Languages
Bill Hersh	Guitar
Bimal Desai	
Pesha Rubinstein	Reading

Right Outer Join

Will include all rows in the right table, display blanks from left

Faculty	
User_Id	User_name
54	Tom Payne
30	Alexis Carter
41	Bill Hersh
29	Bimal Desai
12	Pesha Rubinstein

Interests	
User_ID	Hobby
54	Hiking
30	Languages
41	Guitar
61	Home Improvement
12	Reading

```
SELECT Faculty.User_Name, Interests.Hobby

FROM Faculty

RIGHT OUTER JOIN Interests

ON Faculty.User_ID = Interests.User_ID
```

Faculty	Interests
Tom Payne	Hiking
Alexis Carter	Languages
Bill Hersh	Guitar
	Home Improvement
Pesha Rubinstein	Reading

Full Outer Join

Includes all rows in both tables, blanks in both

Faculty	
User_Id	**User_name**
54	Tom Payne
30	Alexis Carter
41	Bill Hersh
29	Bimal Desai
12	Pesha Rubinstein

Interests	
User_ID	**Hobby**
54	Hiking
30	Languages
41	Guitar
61	Home Improvement
12	Reading

```
SELECT Faculty.User_Name,Interests.Hobby

FROM Faculty

FULL OUTER JOIN Interests

ON Faculty.User_ID = Interests.User_ID
```

Faculty	Interests
Tom Payne	Hiking
Alexis Carter	Languages
Bill Hersh	Guitar
	Home Improvement
Bimal Desai	
Pesha Rubinstein	Reading

Cartesian (Cross) Join

Rarely used – gives you the "cross product" of both tables

Often arises by accident when joins don't have correct constraints or if you omit a WHERE clause constraint

Sometimes used to generate test data

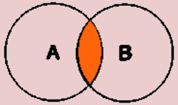
Uses SQL "CROSS JOIN" statement, omits the "ON" statement

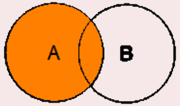
```
SELECT Faculty.User_Name, Interests.Hobby
FROM Faculty
CROSS JOIN Interests
```

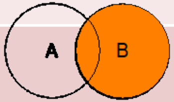
Probably not what you want…
Returns 25 rows.
Can you guess why?

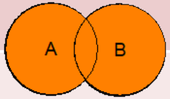
Supplement: SQL Joins as Venn Diagrams

Join Type	SQL Code	Diagram	Explanation

INNER	`select a.*, b.* from a`
	`inner join b`
	`on a.id = b.id`
			Only returns rows present in both tables. No nulls in columns from A or B

LEFT OUTER	`select a.*, b.* from a`
	`left outer join b`
	`on a.id = b.id`
			Returns all of A, regardless of match in B (columns from B may be null)

RIGHT OUTER	`select a.*, b.* from a`
	`right outer join b`
	`on a.id = b.id`
			Returns all of B, regardless of match in A (columns from A may be null)

FULL OUTER	`select a.*, b.* from a`
	`full outer join b`
	`on a.id = b.id`
			Returns all cells from both, including nulls in A and B

Nested Subqueries in SQL

Mimic an "inner join" using nested subquery syntax

Substitute results of a subquery for **"where [column] in"** clause in place of a list

Example: suppose you want all meds ordered for patients between ages 4 and 5. The data are in two tables, a "patients" table and "medications", with "pat_id" as PK in patients, FK in medications

- **First, you identify all patients between 4 and 5 – this is the subquery**
- **Then you pass the results of the subquery as a list of values to the "IN" operator**

```
select * from medications
where pat_id in
          (select pat_id from patients
          where pat_age between 4 and 5)
```

Class Exercise

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Two tables describe the customers and orders for an online shoe store.

Orders.custId is a foreign key that refers to the column **Customers.Id**

Write a query that returns the orderId, name, and city of all customers who ordered “dancing shoes”

Class Exercise

Write your “SELECT” statement first. Since we’re mixing columns from two tables, we need to qualify the table for each column using “dot” notation:

SELECT Orders.orderId, Customers.name, Customers.city

Now we have to choose a “FROM” table – it doesn’t matter which you choose in this example. If you choose Orders, you’ll have to join Customers. If you choose Customers, you’ll have to join Orders. Let’s choose Orders:

FROM Orders

Here comes the join – in this case, we want an “inner” join because we don’t care about nulls on either side (orders without customers or customers without orders). We link the tables on the PK/FK relation specified in the database schema:

INNER JOIN Customers
ON Orders.custId = Customers.Id

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

So far, we have…

```
SELECT Orders.orderID, Customers.name, Customers.city
FROM Orders
INNER JOIN Customers
ON Orders.custID = Customers.Id
```

All we need now is a "where" clause to limit the results to customers who ordered "dancing shoes". **Voila! Our final query**:

```
SELECT Orders.orderID, Customers.name, Customers.city
FROM Orders
INNER JOIN Customers
ON Orders.custID = Customers.Id
WHERE Orders.itemDesc = 'dancing shoes'
```

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

**note the single quotes around text string SQL*

Class Exercise

SELECT Orders.orderId, Customers.name, Customers.city

FROM Orders

INNER JOIN Customers

ON Orders.custId = Customers.Id

WHERE Orders.itemDesc = ‘dancing shoes’

What is the result of this query?

orderId	name	city
66	Pesha	Bethesda

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

SELECT Orders.orderId, Customers.name, Customers.city

FROM Orders

INNER JOIN Customers

ON Orders.custId = Customers.Id

WHERE Orders.itemDesc = 'dancing shoes'

Bonus questions (assume you still want to show the same 3 attributes):

1) Can you modify the query to show only customers from Seattle?
2) Can you modify the query to show only customers from Seattle who did not order "golf cleats"?
3) Can you modify the query to show all orders, including those where the customer is not listed in the Customers table?
4) Can you modify the query to show all customers whose first name starts with the letter "B" (even if they didn't place an order)?

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

Show only customers from Seattle:

SELECT Orders.orderId, Customers.name, Customers.city
FROM Orders
INNER JOIN Customers
ON Orders.custId = Customers.Id
WHERE Customers.city = ‘Seattle’

What is the result of this query?

orderId	name	city
69	Tom	Seattle

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

Show only customers from Seattle who did not order golf cleats:

```
SELECT Orders.orderId, Customers.name,
Customers.city
FROM Orders
INNER JOIN Customers
ON Orders.custId = Customers.Id
WHERE Customers.city = ‘Seattle’
AND Orders.itemDesc <> ‘golf cleats’
```

What is the result of this query?

orderId	name	city
0 rows returned		

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

All orders, including those with missing customer

SELECT Orders.orderID, Customers.name, Customers.city
FROM Orders
LEFT OUTER JOIN Customers
ON Orders.custID = Customers.Id

orderId	name	city
65	Bimal	Philadelphia
66	Pesha	Bethesda
67		
68	Alexis	Atlanta
69	Tom	Seattle

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

Class Exercise

All customers whose first name starts with letter “B”,

even those with no orders:

SELECT Orders.orderID, Customers.name, Customers.city
FROM Orders
RIGHT OUTER JOIN Customers
ON Orders.custID = Customers.Id
WHERE Customers.name LIKE ‘B%’

orderId	name	city
65	Bimal	Philadelphia
	Bill	Portland

Customers

Id (PK)	name	city
1	Alexis	Atlanta
2	Bimal	Philadelphia
3	Tom	Seattle
4	Bill	Portland
5	Pesha	Bethesda

Orders

orderId (PK)	custId (FK)	itemDesc
65	2	flip flops
66	5	dancing shoes
67	7	clogs
68	1	hiking boots
69	3	golf cleats

**use the “LIKE” condition to match strings. Note the “%” which represents a wildcard*

Notes about “LIKE”

Wildcards: “%” matches any length, “_” must match a single character

where lastName like ‘Smith_’

…would match “Smithe”, “Smiths”, “Smithy”

…would NOT match “Smith” or “Smithers”

LIKE is case sensitive, so you may need to case-correct the string before matching

- UPPER([char]) → converts [char] to all upper-case
- LOWER([char]) → converts [char] to all lower-case

This expression:

where lower(lastName) like ‘desa%’

…would match “Desai”, “DeSai”, “desai”, “DeSalles”, etc…

A SQL Developer walks into a bar.

She sees two tables and says,

“Hey, may I join you?”

Your well-earned moment of Zen…

Hierarchical Database

Structurally different from RDBMS

Optimized for rapid transactions of hierarchical data

In very simple terms, makes it easy to know “every attribute about one thing” (quickly retrieve all known information about patient 1001)

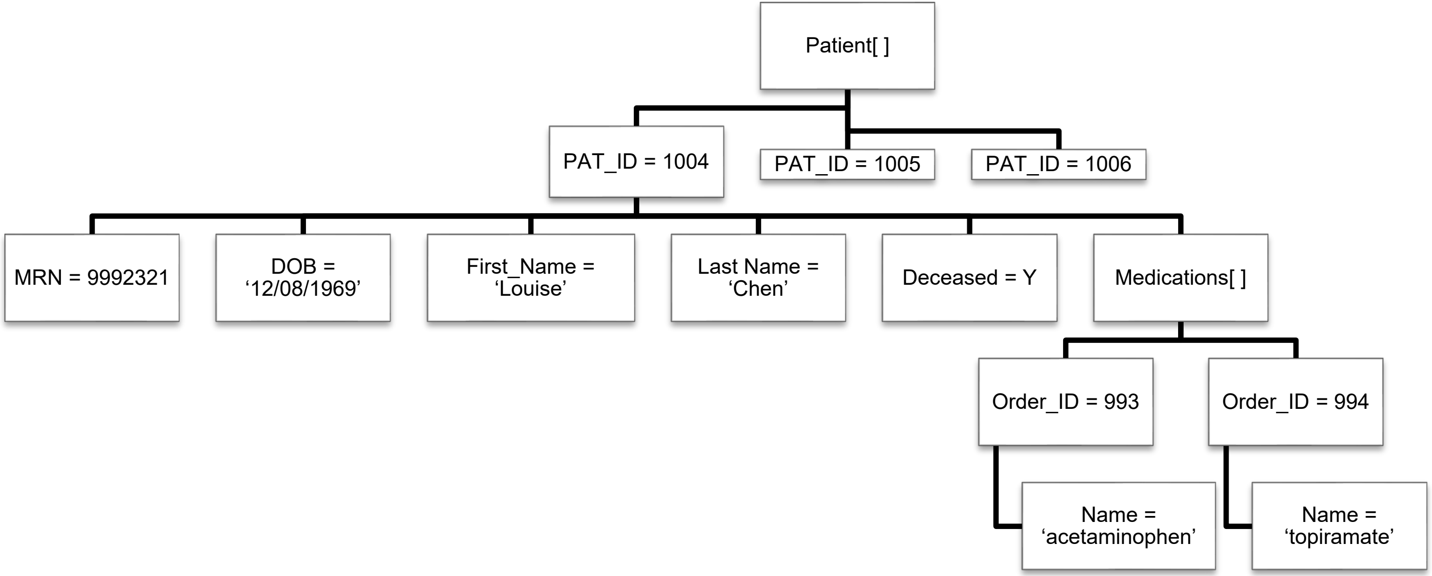
Computationally easy to traverse the tree. Can only traverse tree from root (top parent) node

Ex: “find all deceased patients who were ordered topiramate” would be “easier” in relational DB than hierarchical DB

Child nodes can only have 1 parent

- Difficult to model relationship between child nodes (many-to-many, recursive relationships)

Example Hierarchical DB



History of MUMPS

COMPUTERS AND BIOMEDICAL RESEARCH 2, 469–485 (1969)

Design and Implementation of a Clinical Data Management System*

R. A. GREENES, A. N. PAPPALARDO, C. W. MARBLE, AND
G. OCTO BARNETT

Laboratory of Computer Science,
Massachusetts General Hospital,
Department of Medicine
Harvard Medical School,
Boston, Massachusetts 02114

Received March 10, 1969

Increasing activity in the use of computers for acquisition, storage, and retrieval of medical information has been stimulated by the growing complexity of medical care, and the needs for standardization, quality control, and retrievability of clinical data. Criteria for the design of a clinical data management system include flexibility in its interface with its environment, the capability of handling variable length text string data, and of organizing it in tree-structured files, the availability of this data to a multi-user environment, and the existence of a high-level language facility for programming and development of the system. The scale and cost of the computer configuration required to meet these demands must nevertheless permit gradual expansion, modularity, and usually duplication of hardware. The MGH Utility Multi-Programming System (MUMPS) is a compact time-sharing system on a medium-scale computer dedicated to clinical data management applications. A novel system design based on a reentrant high-level language interpreter has permitted the implementation of a highly responsive, flexible system, both for research and development and for economical, reliable service operation.

History of MUMPS

Described in 1969 by Greenes, Pappalardo, Marble, and Barnett

“MGH Utility Multi-Programming System”

Design goals

- Flexible interface (e.g. lab systems, notes, variable output format)
- Variable length text-handling
- Hierarchical design to support complexity of clinical data and update/retrieval methods
- Multi-user access (original paper recognized potential for conflicting updates, need to have ACID transactions)
- Large storage capacity
- Low CPU usage
- A high-level programming language to make interface design less time-consuming, more efficient

MUMPS renamed “M” in 1993 by M Technology Association, recognized by ANSI in 1995

MUMPS and its derivatives, such as Intersystems Caché, are among the most widely used transactional DBs for EHRs today

Design of MUMPS predated, anticipated the “NoSQL” and “schema-less DB” movement

MUMPS as a Procedural Language

```
→WRITE 1

1.10 READ !,"UNIT NO.   ",X
1.15 IF 'X:3N"-"2N"-"2N TYPE "   ILLEGAL" GOTO 1

→DO 1

UNIT NO.   123-45-678   ILLEGAL
UNIT NO.   12-345-67  ILLEGAL
UNIT NO.   123-456-78   ILLEGAL
UNIT NO.   123-45-67
```

This programming snippet reads user input from teletype at the prompt "Unit No." and assigns the value to variable X

Line 1.15 uses a ternary operator (IF-THEN-ELSE) to validate the format of the string X, in this case, that it's the form of 3 digits, a dash, 2 digits, a dash, and two digits.

If the pattern does not match, it displays the phrase "ILLEGAL" and returns to 1.10

Can you guess what these snippets do?

(hint: imagine this code as the user interface for a lab information system)

```
→WRITE 2,9

2.05 SET DCT="CA,P,FBS,CHOL,TP,NA,K,CL,CO2,SGOT,LDH,VDB,BUN,CRE"
2.10 READ !,"TEST: ",TES
2.20 FOR I=1:1:14 IF $PIECE(DCT,I)=TES QUIT GOTO 2.3
2.25 TYPE " ???" GOTO 2.1
2.30 ASK !,"RESULT= ",RES GOTO I+3
2.40 READ "   PROB. ERROR...OK? ",X IF 'X["Y" GOTO 2.3
2.50 DO 100 TYPE ! GOTO 2.1

9.10 IF RES>160!RES<120 GOTO 2.4
9.20 GOTO 2.5

→DO 2

TEST: MA ???
TEST: NA
RESULT= 125

TEST:
? 2.10 IOINT

→9.1 IF RES>150!RES<130 GOTO 2.4
→DO 2

TEST: NA
RESULT= 125   PROB. ERROR...OK? Y

TEST:
```

MUMPS as a Hierarchical DB

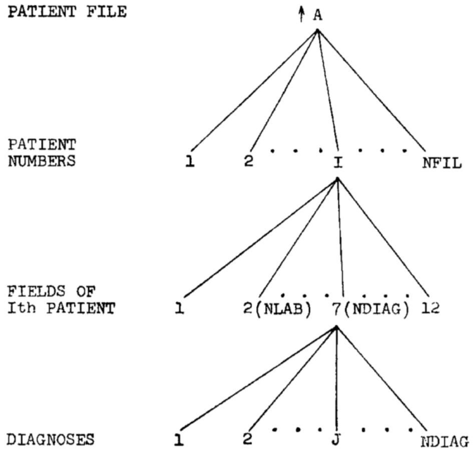


FIG. 5. A simplified tree-structured patient file, stored in a global array, named "↑A", used in the data retrieval program example of Fig. 6. The first level of the array indicates patient identification; the *I*th branch is expanded here to show the structure for the data of patient I. The next level is used to represent class of information. We are interested in field 7, diagnoses; the number of diagnoses present for the *I*th patient is indicated at this level. The diagnosis field is expanded at the third level, which contains a number of individual diagnostic statements.

MUMPS Global Variables

“[H]ierachically organized, symbolically accessed” structure – KEY/VALUE database

Local variables are defined in the scope of the program

Global variables referenced by an up arrow symbol (later became a caret “^”)

This code retrieves a patient in the Active Patient Record (APR) global that matches a local variable “UN” (hospital unit number, or location of patient) and assigns the name and age:

```
SET ^APR(UN, NAME)=“DOE, JOHN”, ^APR(UN, AGE)=“34”
```

This code traverses a patient’s record UN→CHEM→N (unit number, chemistry results, sodium), and assigns it a string value, concatenated from two local variables DATE and TEST:

```
SET ^APR(UN,CHEM,N)=$DATE.”,”,TEST
```

Object Databases

Data represented as data objects

Support for more data types (graphics, photo, video, webpages)

Object DBs are usually integrated into programming language, so accessing data doesn't require complex driver configuration

Increased use recently with development of web applications, most web application frameworks support interaction with OODBMS

Commercial example: Intersystems Caché – the OODBMS behind the Epic EHR

Unified Modeling Language

Standard toolset for describing aspects of databases, software, business processes

Class diagram to describe OO classes (name, hierarchies, attributes, methods)

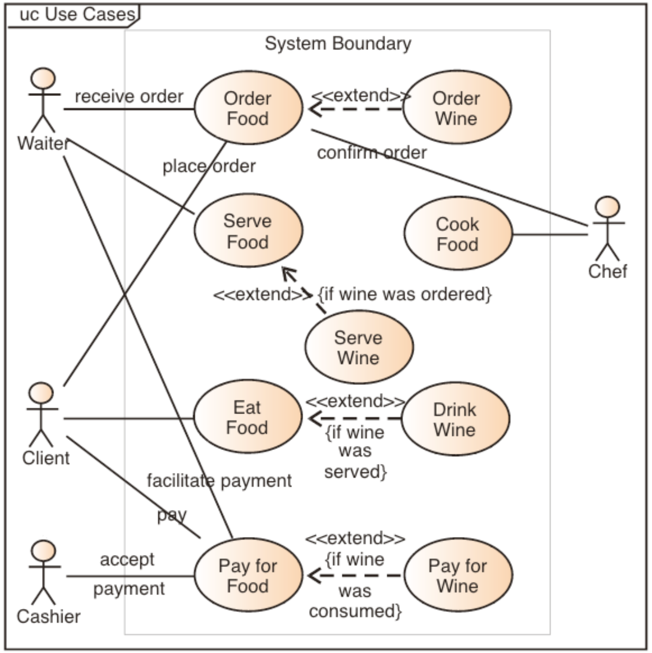
Activity diagram ~ process flowchart, stepwise description of decisions, consequences, inputs, outputs

Unified Modeling Language

Use Case diagram – describes actors, goals, dependencies

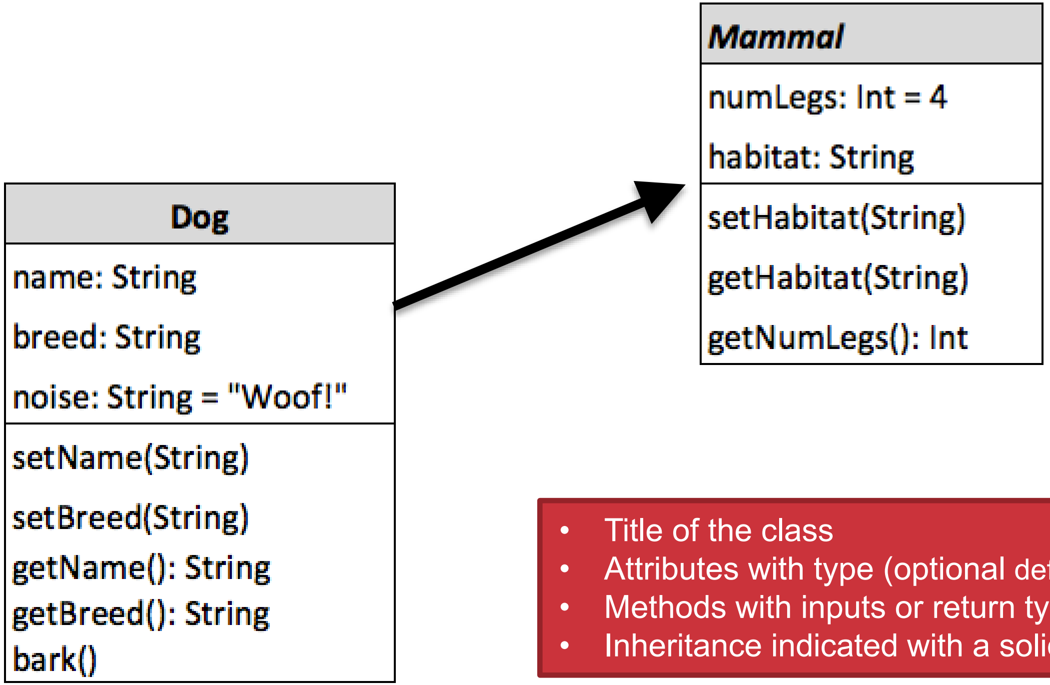
Entity-Relationship diagram – describes objects and their relationships

ER diagram can then be used to define RDBMS logical schema, which DB programmers can use to build physical schema of a DB



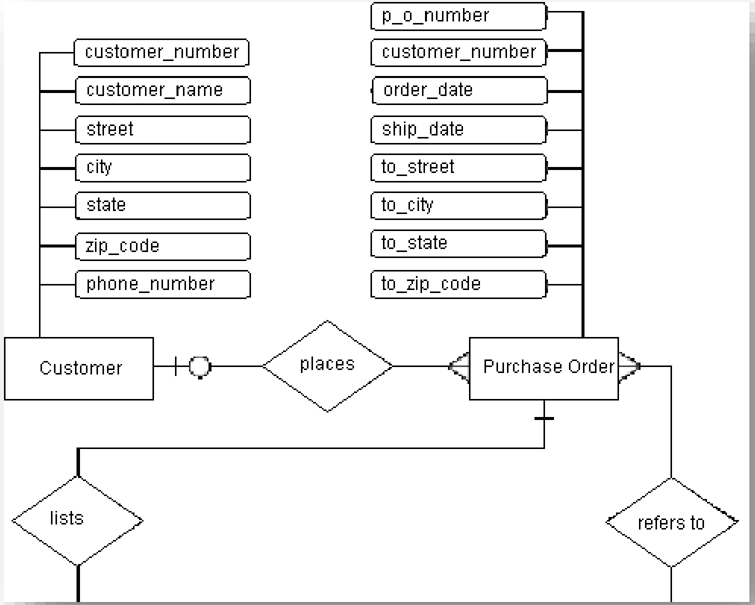
(Image credit: https://en.wikipedia.org/wiki/File:Use_case_restaurant_model.svg)

UML Class Diagram



- Title of the class
- Attributes with type (optional default values)
- Methods with inputs or return types
- Inheritance indicated with a solid arrow pointing to parent class

UML E-R Diagram



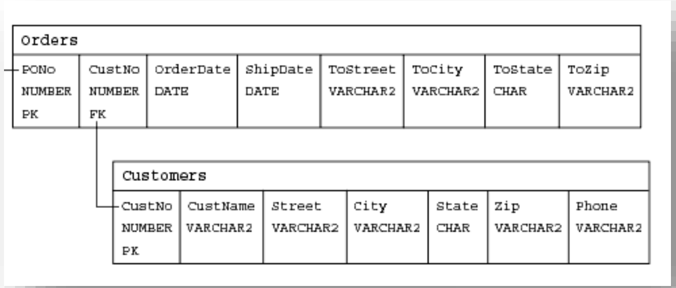
"E-R" = Entity Relation

Note relationship between Customer and Purchase Order:

- A customer is an optional participant (the "O" symbol)
- Only one customer can participate (the "|" symbol)
- A customer can have multiple purchase orders (the three-pronged arrow symbol)
- Details the attributes of Customer and Purchase Order

Source: https://docs.oracle.com/cd/B13789_01/java.101/b12021/dev.htm

UML E-R Diagram




```
CREATE TABLE Customers (
  CustNo   NUMBER(3) NOT NULL,
  CustName VARCHAR2(30) NOT NULL,
  Street   VARCHAR2(20) NOT NULL,
  City     VARCHAR2(
  State    CHAR(2) N
  Zip      VARCHAR2(
  Phone    VARCHAR2(
  PRIMARY KEY (CustN
);
```

```
CREATE TABLE Orders (
  PONo      NUMBER(5),
  Custno    NUMBER(3) REFERENCES Customers,
  OrderDate DATE,
  ShipDate  DATE,
  ToStreet  VARCHAR2(20),
  ToCity    VARCHAR2(20),
  ToState   CHAR(2),
  ToZip     VARCHAR2(10),
  PRIMARY KEY (PONo)
);
```

Based on the E-R Diagram, a developer can:

- describe the logical schema for the database
- create physical schema and DDL/SQL code to create tables
- create object classes that map to database tables
- map object classes to DB tables using an ORM tool

Source: https://docs.oracle.com/cd/B13789_01/java.101/b12021/dev.htm

Reliable DB Transactions: the ACID Test

Atomicity – transaction is indivisible, it either happens or it doesn't, no possibility of a partial transaction (ex: a DB transaction that updates 2 cells – it either does both or neither)

Consistency – transaction meets all constraint rules (can't add a DATE to an INT field, can't have a non-unique PK)

Isolation – RBDMS must be able to sequence simultaneous transactions (ex: 2 transactions to update the same cell. Both must take place, but not at same time, or else you have a write-write failure)

Durability – system must be tolerant to failure (ex: RDBMS has queued 200 transactions in memory, and power fails. How do you know if all 200 transactions took place?)

Normalization

Techniques of structuring tables to reduce redundancy, dependency between tables

Consider the “Flat File” example from earlier

- One of the “Medication” cells has 2 entries
- This violates a rule known as 1st Normal Form (1NF)

Goals of Normalization

- To free the collection of relations from undesirable insertion, update, and deletion dependencies
- To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the lifespan of application programs
- To make the relational model more informative to users
- To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by

See SUPPLEMENTAL MATERIALS for an overview of the most common Normal Forms

Normalization & Denormalization

The “Normal Forms” were described by Codd and Boyce, who described techniques to reduce inconsistencies and dependencies in relational databases. These forms are named numerically 1NF, 2NF, 3NF, BCNF, 4NF, 5NF, and 6NF.

For a practical tutorial on database normalization, see here: https://www.guru99.com/database-normalization.html

In practice, a database that is in **Third Normal Form (3NF)** can be called “normalized”

There are higher forms of normalization beyond 3NF, like “Boyce-Codd Normal Form” (abbreviated “BCNF”)

Normalized DBs are safe against most INSERT, UPDATE, and DELETE anomalies, however, to generate a report, you have to “denormalize” the data – requires lots of PK & FK “JOIN” logic in your query

For high-performance RDBMS apps, **denormalized** schema may be preferable to allow single-table lookup functions with an index, to avoid additional JOINs and full-table scans

Also, you need to denormalize data to aggregate the data into meaningful groups or reports (eg: all meds for patient X)

That is often the role of reporting tools, analytics, data marts, etc.

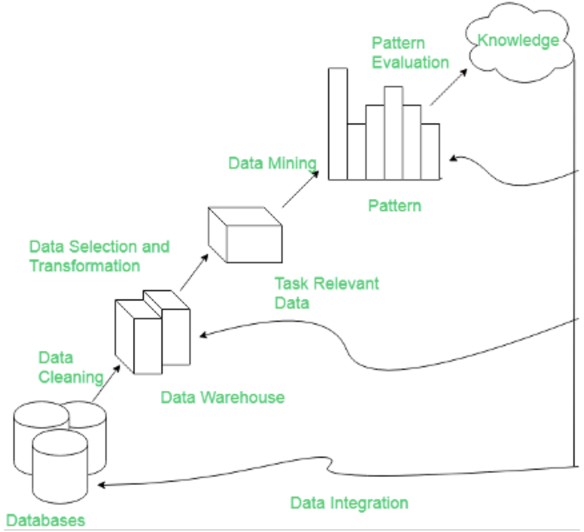
Ex: For performance and functionality, data marts are intentionally denormalized

Data Mining & Knowledge Discovery (KD)

Data Mining: automatic summarization, identifying essential information, discovery of patterns in data

Three key steps:

1. Pre-processing – clean, integrate, select, transform
2. Extraction
3. Evaluation & Presentation



Source: KDD Process in Data Mining

Data Mining & KD: Terms to Know

•**Data Cleaning**: Removal of noisy and irrelevant data from collection.

- Addressing missing values, noisy data, data discrepancy through various data transformations

•**Data Integration**: Combining heterogeneous data from multiple sources into a common source (DataWarehouse).

- Migration, synchronization, and ***ETL***(Extract-Load-Transformation) process.

•**Data Selection**: Data selection is defined as the process where data relevant to the analysis is decided and retrieved from the data collection.

- Can involve statistical methods to identify patterns / outliers, clusters (regression, machine learning)

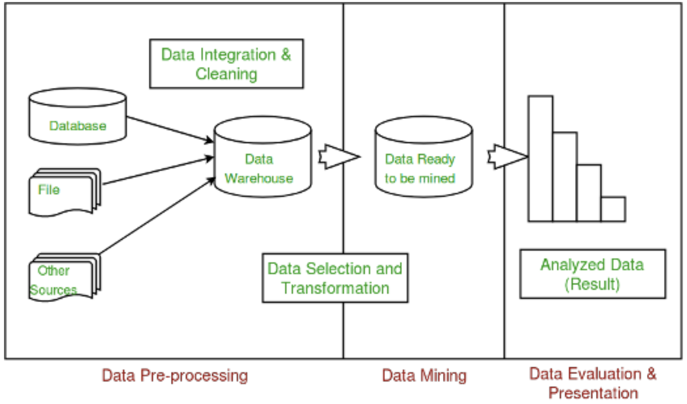
•**Data Transformation**: Data Transformation is defined as the process of transforming data into appropriate form required by mining procedure. Data Transformation is a two-step process:

•***Data Mapping***: Assigning elements from source base to destination to capture transformations.

•***Code generation***: Creation of the actual transformation program.

•**Pattern Evaluation:** Identify interesting patterns, summarize & visualize findings

•**Knowledge Representation:** generation of reports, tables, rules



[Source: Data Mining]

Data Mining & KD - Example

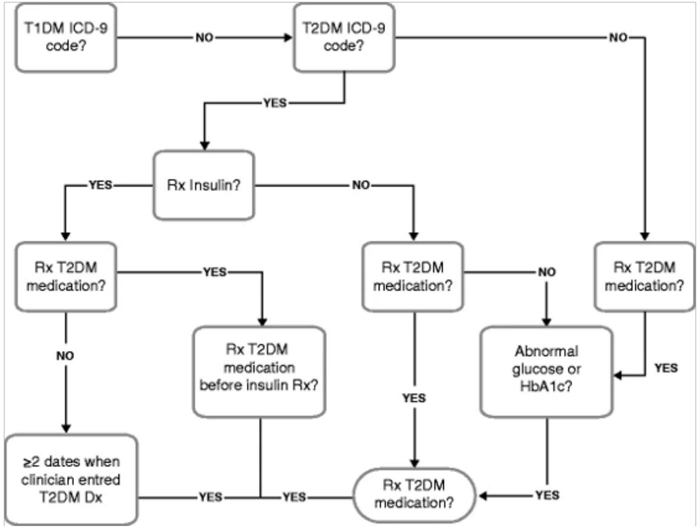
Task: use EMR data to identify patients with a specific disease phenotype for correlation with genomic data

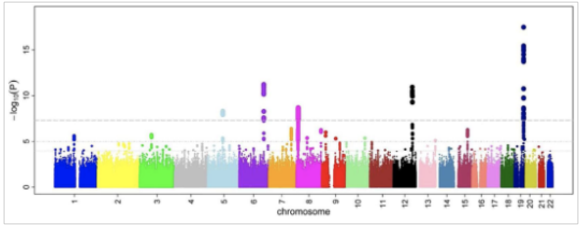
•**Data cleaning & integration** – combing administrative, clinical, and genomic data

•**Data selection** – machine learning to "train" an algorithm to identify suspected cases

•**Data transformation** – NLP tools to extract medication mentions in free-text notes, mapping to canonical terminology

•**Pattern evaluation, Knowledge Representation** - "Manhattan plot" to highlight which SNPs are associated with candidate disease





Source: https://en.wikipedia.org/wiki/Manhattan_plot#/media/File:Manhattan_Plot.png

Credit: Wei, WQ., Denny, J.C. Extracting research-quality phenotypes from electronic health records to support precision medicine. Genome Med 7, 41 (2015). https://doi.org/10.1186/s1307 -015-01 -y

Data Warehouse & Data Marts

Extract/Transform/Load (ETL) process gets transactional data into a format that is optimal for reporting / queries

Real life example: Epic EHR runs on Intersystems Caché Object DB for transactional processing. Has a nightly process to push data into an RBDMS (e.g. Oracle SQL)

Datamart is a smaller collection of related tables and data derived from the warehouse for a specific purpose, usually for analysis, report generation, spreadsheet, dashboards, etc.

Example: EHR may have a real-time transactional DB, nightly dump to a SQL data warehouse, and weekly extracts to a datamart to generate an updated enterprise asthma performance dashboard.

Data Storage Strategies: Terms to Know

Location:

- **On premises** "on prem" - you own and operate the data center
- **Co-location** – you lease a data center and manage the servers, but not the facility
- **Cloud** – out outsource the data center and server management, but you manage the database itself

Storage Medium:

- Hard drive (cheap, but slower and prone to failure w/ moving parts) - often put into Redundant Array of Inexpensive Disk (RAID) with various techniques for redundancy such as mirroring, parity checks
- Solid state drives (expensive, smaller, faster)
- Tape or other medium

Tradeoff of durability vs redundancy cost vs speed

- Ex: AWS "S3 Glacier Deep Archive" vs. "S3 Standard"
- Both highly-available, but Glacier is designed for very infrequent access – slow (minutes/hours), very inexpensive. S3, in contrast, has latency of milliseconds and can be used for real-time production applications

Data Governance / Stewardship

Data Governance: Framework of processes, tools, methods, and oversight that ensures availability, usability, consistency, integrity, and security of data. Benefits of strong DG practices include improved ability to collect, view, store, exchange, aggregate, analyze, manage, archive, and reuse data.

Data Steward: Ensures that data governance processes are followed and enforced.

FAIR Acronym: Findable (e.g., via metadata), Accessible, Interoperable, Reusable

Governance/Stewardship Examples

- What is the definition of "length of stay" within your organization?
- What controls do you have in place re: access to clinical data warehouse?
- Does your organization have a policy on cloud storage?
- By what process are data for public reporting reviewed for accuracy / consistency?
- What happens if an organizational data element changes (e.g. CDC changes definition of "blood stream infection" or EHR vendor changes underlying table structure)?
- How do you request enterprise financial data and who has access to it?

Example Data Governance Tools

SPECIALTY

Medication Safety

Download

Quality & Safety > Patient Safety

Name	Title	Email	Phone
Data Stewards:			
Stewardship Type: General			
[illegible]	Pat Safety & Qlty Data Sr An	[illegible]	
Technical Owners:			
Stewardship Type: General			
[illegible]	Pat Safety & Qlty Data Sr An	[illegible]	
Data Sponsors:			
Stewardship Type: General			
[illegible]	Sr. Med Dir Patient Safety	[illegible]	

Metrics:

Medication Reconciliation Performance

Medication Serious Safety Event Rate - Rolling 12 Month Average

Preventable Adverse Drug (Medication) Events

Medication Reconciliation Performance

Quality & Safety > Patient Safety > Medication Safety

% of patients that have had a completed medication reconciliation in Epic during the first 24 hours of their admission

Focus Area Category:
Medication Safety

Data Stewards (Primary Contact):
[illegible]

Technical Owners:
[illegible]

Data Sponsors:
[illegible]

Calculation:	Total number of patients that had a medication reconciliation done / Total number of patients admitted
Desired Direction:	Up
Type:	Percent
Benchmark:	Joint Commission on Accreditation of Healthcare Organization (JCAHO)
Source:	EPIC Medical Reconciliation Dashboard + Chart Review
Inclusions:	All inpatient transfers, Medication reconciliation done within 24 hours
Exclusions:	Ambulatory and outpatient admissions
Data Elements:	1. Total number of patients with completed medication reconciliation in EPIC in the first 24 hours of admission, 2. Total number of all patients admitted

Associated Data Assets:

Cohorts:

No associated cohorts identified at this time. If you would like to provide us with this information, please contact DataGovernance@email.chop.edu.

− QlikViews (2):

CHOP Performance Metrics

Medication Reconciliation Dashboard

Network Topologies

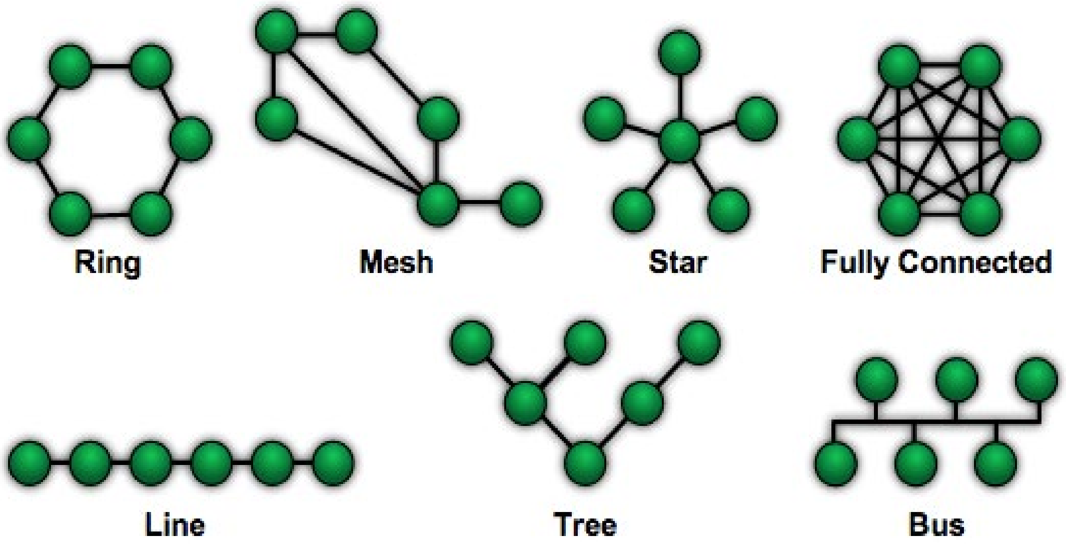


Image credit: https://upload.wikimedia.org/wikipedia/commons/9/97/NetworkTopologies.svg

Network Topologies

Patterns of links between elements of a computer network

Choice of topology determines fault tolerance, redundancy, and scalability

Phone telephony is an example of a point-to-point connection, so is connection between your CPU and the hard-drive

Centralized – Star, Tree

Decentralized – Mesh, Fully Connected

Features of Network Protocols

"Handshake"

Acknowledgement

Payload

Multiple protocols exist for multiple purposes

Distinguish the Network Protocol from the Data or Encoding Standard

- Ex: HL7 v2.x message is a pipe-delimited text file, v3 is an XML file, both can be transmitted via TCP/IP across a network

The TCP/IP Stack

Network protocol used for internet communications

TCP = transmission control protocol

IP = internet protocol

UDP (user datagram protocol) is an alternative to TCP

- **Key differences**
  - **TCP requires acknowledgement, UDP does not**
  - **TCP guarantees sequence/order of packets, UDP does not**
- TCP used where packet loss is unacceptable (will resend until acknowledgement or timeout)
- UDP used where packet loss is less important (Voice over IP aka “VoIP” or streaming protocols)

Did you hear the joke about the TCP packet that walked into a bar?

Packet beer give UDP bar me says bartender walks into a.

The best thing about telling a UDP joke is that you don’t care if anyone gets it.

OSI Seven Layer Model

Host

- Data [transmission of message via HTTP, SMPT, FTP]

  7. Application (HTTP is an application layer protocol)

  6. Presentation / Syntax (e.g. character encoding in ASCII or data encryption)

  5. Session (ex: a web conference may have a persistent session to synch audio and video)

- Segments

  4. Transport [transmission of **segments** via TCP, UDP]

Media

- TCP Packet / UDP Datagram

  3. Network [transmission of **packets** via IP, DNS server, through routers]

- Frame

  2. Data Link [transmission of **frames** via Ethernet or PPP]

- Bit

  1. Physical [transmission of binary bits via copper wire, coaxial, or fiber optic cable]

Examples of Protocols by OSI Layer

Layer	Example Protocols
7: Application	**POP3/IMAP4** for email, **HTTP** for web content, **FTP** for file transfer, **SSH** and **HTTPS** for secure browsing
6: Presentation	Encryption, decryption, conversion to character sets (like ASCII)
5: Session	**LDAP** “Lightweight Directory Access Protocol” for authenticating users against X.500 directories
4: Transport	**TCP** “transmission control protocol” (with acknowledgement), **UDP** “User Datagram Protocol” (without acknowledgement)
3: Network	**IPv4, IPv6, DHCP** “dynamic host control protocol” used to assign IP addresses to hosts, for example, when you connect to a wireless hotspot
2: Data Link	**ARP** – “address resolution protocol” used by TCP to communicate with hosts when only neighboring hosts’ addresses are known
1: Physical	none

OSI Real World Analogy

(adapted from http://som.csudh.edu/fac/lpress/471/hout/netech/postofficelayers.htm)

7: Application Layer – I send a letter to Bob. Bob receives and opens the letter.

6: Presentation Layer – my letter is encoded in roman script using the English language

5: Session Layer – I may include one or more letters per session (envelope) as long as I have appropriate postage

4: Transport Layer – There is a typo in the address. The postal service marks it "recipient unknown" and sends it back. I get details of failure or confirmation of success (ex: registered mail)

3: Network Layer – physical mail is sent by plane between cities. Pilot has no awareness of the the final destination of the letter she is carrying

2: Data Link Layer – Postal Service worker drives truck within city to deliver message

1: Physical Layer – my letter is comprised of ink or graphite on a piece of paper, folded and tucked into an envelope

Telecommunications

Telephony has changed rapidly in past decade

Popularization of mobile, wifi, and VOIP

Video conferencing

- Web conferencing / collaboration via H.264 Scalable Video Coding (SVC)
  - This is the the same codec*, known as MPEG-4, used for distribution of video content on IP, like YouTube

*CODEC = coder / decoder – a compression algorithm used for a digital stream to transmit audio, video, etc. They can be “lossy” or “lossless”. Lower bitrate often means lower fidelity.

Short Range Wireless Standards

Short Range PAN - Personal Area Network

- **RFID** (one way) and **NFC** (two way)
- **IEEE 802.15** – Wireless Personal Area Network and derivatives (Bluetooth & Infrared Data Association or IrDA)

Medium Range Wireless Standards

Medium Range WLAN – Wireless Local Area Network

- **802.11b** – Max 11Mbps, interferes with other 2.4Ghz devices like microwaves, Bluetooth, cordless phones. Popularized WiFi
- **802.11g** – Max 54 Mbps, same band as 802.11b, same interference concern.
- **802.11n** – uses both 2.4Ghz and 5 Ghz spectrum for max speeds of 54 Mbps and 600 Mbps respectively. Speed enhanced by MIMO (Multiple Input, Multiple Output)
- **802.11ac** – standard for “gigabit wifi” – 1 Gbps

Long Range Wireless Standards

Alphabet Soup & Confusing Marketing Alert!

- WiMax
- CDMA
- 3G
- 4G / 4G LTE (30-50Mbps)
- 5G → emerging standard promising hundreds of Mbps over cellular. Could replace WiFi (e.g. you place a 5G receiver near the window of your house, connect that to a residential router)

Wireless Applications

IP Telephony (“Vocera” devices in healthcare)

SMS text messaging

Various “secure” texting solutions (HIPAA compliant)

RFID/NFC tagging of medical devices, patients

Bluetooth Standard

Historically maintained by IEEE as 802.15.1, but now maintained by Bluetooth SIG

Bluetooth 4.0 standard introduced Bluetooth Low Energy (aka Bluetooth Smart or Bluetooth LE)

Bluetooth LE has recently become very popular in health and fitness

- Healthcare-specific profiles for blood pressure, thermometer, glucose monitor, continuous glucometry
- Fitness-specific profiles for weight scale, running/cycling speed, heart rate, etc.

Bluetooth Low Energy (BLE)

aka “Bluetooth Smart”

Low battery consumption, limited need for data transfer

One-way communication in close proximity

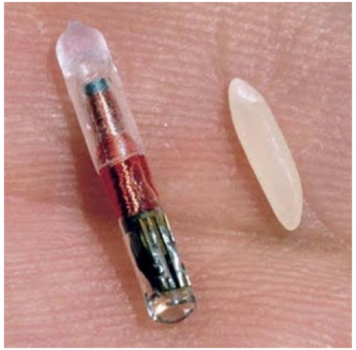
BLE Beacons broadcast packets of data at regular intervals, and devices (like Smartphones) pick them up, detected by pre-installed apps or services.

Uses: indoor navigation, proximity-based marketing

RFID

RFID = Radio Frequency Identification

- 3 “flavors”: passive, active, battery-assisted passive
- Passive relies on power from the reader, but reader has to emit 1000x stronger signal
- Tags are read-only or read-write
- RFID reader sends a signal to interrogate tag
- RFID tag/chip responds with ID and other info
- Like tags, readers can be active or passive
- Uses: animal tags, “Smart cards,” asset tracking



Implantable veterinary RFID chip
Image courtesy Wikimedia

NFC

NFC = Near Field Communication

- Used to establish communication between two electronic devices
- NFC tags passively store data, some can be written to by an NFC device.
- Typical uses = phone-enabled payment (credit card information), PIN storage

Healthcare Applicability & Challenges

Retrofitting older facilities with equipment

Bandwidth limitations as amount of data increases

- MRI wrist = 5MB
- CT 3D reconstruction skull = 120MB
- CT angiogram = 230MB
- Human genome = 850MB (I've seen stats as high as 1.5GB)
- Challenges with compression, image quality, and transmission

Keeping up with demand – more and more "ologies"

Network security – distinct wireless networks for telephony, hospital applications, guest applications. VPN and Remote access

Both RFID and NFC pose "skimming" concerns

"Bring Your Own Device" – everyone has a personal device they'd like to use at work

Network maintenance – uptimes become more critical, and downtimes become dangerous

End of Lecture

Supplement: First Normal Form

A DB is said to be in 1NF if it can meet the following conditions:

- **Each cell contains a single value** (our "Flat File" example breaks this rule)
- **Each record is unique** (no duplicate rows)

Patient_ID	Medication	Pharmacy	Formulary
1001	acetaminophen	Glenbrook	yes
1002	amoxicillin	Medstar	yes
1002	fluticasone	Medstar	yes
1003	cetirizine	Brighton	no

- 1NF is susceptible to certain INSERT, DELETE, and UPDATE anomalies:
  - We can't use Patient_ID to uniquely identify each row – this table requires a **"composite key"**
  - INSERT: A patient can't have a Pharmacy without a prescription, unless we create a row with NULL values
  - DELETE: If the med "acetaminophen" is deleted, the Pharmacy "Glenbrook" ceases to exist
  - UPDATE: If the "Medstar" pharmacy chain changes their name, we have to edit multiple cells in this table

Supplement: Second Normal Form

A DB is said to be in 2NF if it can meet the following conditions:

- **The table must be in 1NF** and…
- **The table must have a single-column, non-composite, primary key**

Event_ID (PK)	Patient_ID	Medication_ID	Pharmacy_ID	Formulary
20131	1001	9991212	704	yes
20132	1002	9984021	216	yes
20133	1002	9933333	216	yes
20134	1003	9906761	844	no

- 2NF is susceptible to certain INSERT, DELETE, and UPDATE anomalies:
  - INSERT: We can't indicate a patient's pharmacy unless there is a medication prescribed
  - DELETE: If you delete the last row that contains med "9906761", you no longer know if it's on formulary
  - UPDATE: to update the formulary status for a Medication_ID may require updating multiple rows

Supplement: Third Normal Form

A DB is said to be in 3NF if it can meet the following conditions:

- **Table meets all criteria for 1NF AND 2NF AND**
- Table must have NO **"transitive functional dependencies",** meaning – changing the value of one cell should not require a change to another row. (In the 2NF example, note that changing the value of a medication ID could require a change to the "Formulary" attribute)

Event_ID	Patient_ID	Medication_ID
20131	1001	9991212
20132	1002	9984021
20133	1002	9933333
20134	1003	9906761

Patient_ID	Pharmacy_ID
1001	704
1002	216
1003	844

Medication_ID	Formulary
9991212	yes
9984021	yes
9933333	yes
9906761	no

- Even 3NF is susceptible to certain INSERT, DELETE, and UPDATE anomalies!
- Example: What if there was a registration error and patient 1001 and patient 1003 are actually the same patient? How can you avoid changing multiple cells in the first table?

Suggested Additional Reading

https://en.wikipedia.org/wiki/Database_normalization

Elmasri R, Navathe SB. Fundamentals of Database Systems. 4th Ed. Addison Wesley; 2004. 1009p.

Kurose JF, Ross KW. Computer Networking: a top-down approach featuring the internet. 3rd Ed. Boston: Pearson; 2005. 821p.

Date CJ, Darwen H. A Guide to SQL Standard (4th Edition). 4th edition. Addison-Wesley Professional; 1996.

Greenes RA, Pappalardo AN, Marble CW, Barnett GO. Design and implementation of a clinical data management system. Comput Biomed Res Int J. 1969 Oct;2(5):469–85.

Resources for Self-Directed Learning

Two great resources for learning SQL online, in your browser

https://www.w3schools.com/sql/

https://sqlzoo.net/

Database normalization:

https://www.studytonight.com/dbms/database-normalization.php