E/R Mapping
Entity: real world object, defined by entity type
Entity type: rectangles. Can be weak (can only be identified by considering the primary key of owner, implies total participation)
Attributes: ovals, attached to entity types and relationships by lines. Can be key (underlined), composite
Relationships: diamonds, lines with labelled multiplicities. May relate entities and other relationships. Can be recursive, totally participative (bold line, when every entity in the connected type participates in at least one of our relationships)
ISA hierarchies: triangles, cause attribute, relation inheritance
Aggregation: enclosing box for relating entire entity set

Relations
Instance (data) and schema (names and types of columns)
Domains are sets of values: domains must be atomic
Relation arity = columns
Relation cardinality = rows
Never contain duplicates

Entity Mapping
Weak entities become entities which contain a foreign key to their owner entity
ISA hierarchies can become 3 relations (1 containing common attributes) or 2 (containing all)

Relational Algebra
σ_C: selection with conditions C
π_A: projection with attributes A (nb: must eliminate duplicates)
$\rho_{A:=B}$: rename A to B in schema
Set theoretic union, intersection, difference, product

Derived Operators
$R_1 \bowtie R_2 = \sigma_C(R_1 x R_2)$ with conditions C: commonly equality
Equi-join: theta join where the condition is field name equality
Natural join: equi-join on all common fields, where the duplicate fields are removed
$A/B = \pi_x(A) - \pi_x((\pi_X(A)xB) - A)$: set of a in A such that for every y in B there is a (x,y) in A

Domain
Queries have the form

Relational Calculus
$\{<x_1,...,x_n> \mid F(x_1,...,x_n)\}$: answer is the tuples that make F true
E.g. names of sailors who reserved all boats:
$$\{<N> \mid \exists I,R,A.<I,N,R,A> \in Sailors \wedge \forall B,C.(\neg(<B,C> \in Boats) \vee (\exists <SI,BI,D> \in Reserves.I = SI \wedge BI = B))\}$$

Tuple Relational Calculus
Variable range over tuples rather than field values: dot notation accesses attributes
E.g. names of sailors who reserved all boats:
$$\{P \mid \forall S \in Sailors.\forall B \in Boats.\exists R \in Reserevations. R.bid = B.bid \wedge S.sid = R.sid \wedge P.name = S.name\}$$

Relational Complete.
Language is relationally complete if it can express all the queries of the relational algebra
Can encode relational algebra in the relational calculus, not other way
Safe queries always have a finite answer: undecidable condition

SQL
LIKE: _ one char, % zero or more
Arithmetic in SELECT, WHERE
UNION, INTERSECT, IN

```
CREATE TABLE Students
(sid CHAR(20), name CHAR(20) LIKE
'[A-Z]%', age INTEGER NOT NULL
BETWEEN 18 AND 120, PRIMARY KEY
(sid), UNIQUE(name, age), FOREIGN
KEY (sid) REFERENCES StudentsToo,
CHECK(name) IN ('Bob', 'Fred');
SELECT * FROM R LEFT OUTER JOIN
Students ON R.sid = Students.sid
CREATE VIEW E(sid, sname) AS SELECT
sid, sname FROM S WHERE rating > 9
```

Bag Semantics
$\{1,2\} \cap \{2,3\} = \{1,2,2,3\}$
$\{1,1,2\} \cap \{1,2,3\} = \{1,2\}$
$\{1,2,1\} - \{1,2,3,3\} = \{1\}$

Algebra Extensions
δ: duplicate elimination
τ_L: sort lexicographically by attributes L; returns *list*
γ_L: group by attributes in L and then introduce attributes for each aggregate function, remove dups
Extend projection with arithmetic

NULL
Outer joins pad columns with NULLs for tuples that don't join
⟗: full, ⟕: left, ⟖: right
Left return all tuples from left table, some from right

Functional Dependency
$X \rightarrow Y$ if attributes in set Y are determined by those in X
Relation instance satisfies FD if

$\forall t_1 t_2 \in R.t_1.X = t_2.X \rightarrow t_1.Y = t_2.Y$
F^+ is the set of FDs logically implied by F (closure)

Armstrong's Axioms Show how to construct the closure of a FD set. Both sound and complete

Reflexivity: $Y \subseteq X \rightarrow (X \rightarrow Y)$

Augmentation: $(X \rightarrow Y) \rightarrow (X, W \rightarrow Y, W)$

Transitivity: $(X \rightarrow Y) \wedge (Y \rightarrow Z) \rightarrow (X \rightarrow Z)$

They have these consequences:

Union: $(X \rightarrow Y) \wedge (X \rightarrow Z) \rightarrow (X \rightarrow Y, Z)$

Pseud-trans: $(X \rightarrow Y) \wedge (W, Y \rightarrow Z) \rightarrow (X, W \rightarrow Z)$

Decompose: $(X \rightarrow Y) \wedge (Z \subseteq Y) \rightarrow (X \rightarrow Z)$

Heaths Rule $R = \pi_{A,B}(R) \theta_A \pi_{A,C}(R)$

Minimal Cover FD set is minimal if every FD has a single element Y, no FD can be removed without losing equality and no FDs X can be shrunk without losing equality

Attribute Closure For X→Y, grow consequences of X, adding Y' if X'→Y' ∈ F for X' ⊆ X. Now F ≤ X→Y iff Y ⊆ X^+

Candidate Key For $R(A_1{:}T_1,\ldots,A_n{:}T_n)$ with FDs F, X is a candidate key for R if $X \rightarrow A_1,..,A_n \in F^+$ and no proper subset of X is a candidate key

De-composition A collection of (relation, query) pairs and Q_0 so that the queries can retrieve the relation from R and Q_0 can retrieve R from pairs $\{R_1,\ldots,R_k\}$ is a lossless join decomposition wrt F if for all r satisfying F, $\pi_{R1}(r) \bowtie \ldots \bowtie \pi_{Rk}(r) = r$

Dependency Preserving Projection of FD set F onto Z:
$F_z = \{X \rightarrow Y \in F^+ \wedge X \cup Y \subseteq Z\}$
Decomposition $\{R_1,\ldots,R_k\}$ dependency preserving if
$F^+ = (F_{R1} \cup \ldots \cup F_{Rk})+$

Normal Forms Prime attributes appear in candidate keys, superkeys are supersets of candidate keys

First NF Domain of all attributes must be atomic (violated by e.g. table w/ student IDs list attribute)

Second NF Partial FD X→Y if for some A∈X, (X-{A})→Y (i.e. not minimal)
Every non-prime attribute is not partially dependent on any key

Third NF If for all $X \rightarrow A \in F^+$ then either A∈X, X is a superkey or A is a member of some candidate key

BCNF If for all $X \rightarrow A \in F^+$ then either A∈X or X is a superkey: unlike 3NF it not necessarily possible

Transaction Set of physical operations that form one logical operation

ACID Atomicity: need log
Database consistency: every transaction sees consistent DB, follows from transaction AI+C

Schedule List of actions from a set of transactions
Well formed: schedule actions in same order as in transaction
Complete: contains commit or abort action for all transactions
Serial: actions of transactions not interleaved, used for proof

WR Conflict T2 reads a database object modified by uncommitted T1

RW Conflict T2 changes the value of an object read by in-progress T1

WW Conflict T2 writes to object already written by in-progress T1

S2PL Obtain shared S lock before reading, exclusive X lock before writing,released when complete, phases independent, commutative only needs shared lock if object not read back

OLAP Historical database for analysis
Mostly reads, optimize schema for query processing
ROLAP: backed by relational DB
MOLAP: multidimensional DB

Data Cube Multidimensional representation of data, e.g. sales by dimensions product, date, country
Can include aggregates in cube, e.g. sales sums at edges

Data Warehouse Subject-oriented, integrated, time-variant, non-volatile

Operations Roll up, drill down: move between high/low level summary
Concept hierarchy navigation: all → (Europe → (Germany|Spain) | North America → (Mexico|..))

Schema Star: fact table in the middle connected to dimension tables (e.g. store ID to name mapping)
Constellation: dimensions tables normalised into further stars

Cube: breaks relational model, possibly N-dimensional. Operators like sale(*,*,*): total sales, sale(c2,p2,*): sum of sales data for c2, p2 over time