Tuesday, October 25, 2022 12:23 PM

Database Concepts

Roles in the Intergrated System

Sophisticated Casual
- trained in DB languages
- Access data directly (queries)
- Auth given by DBA

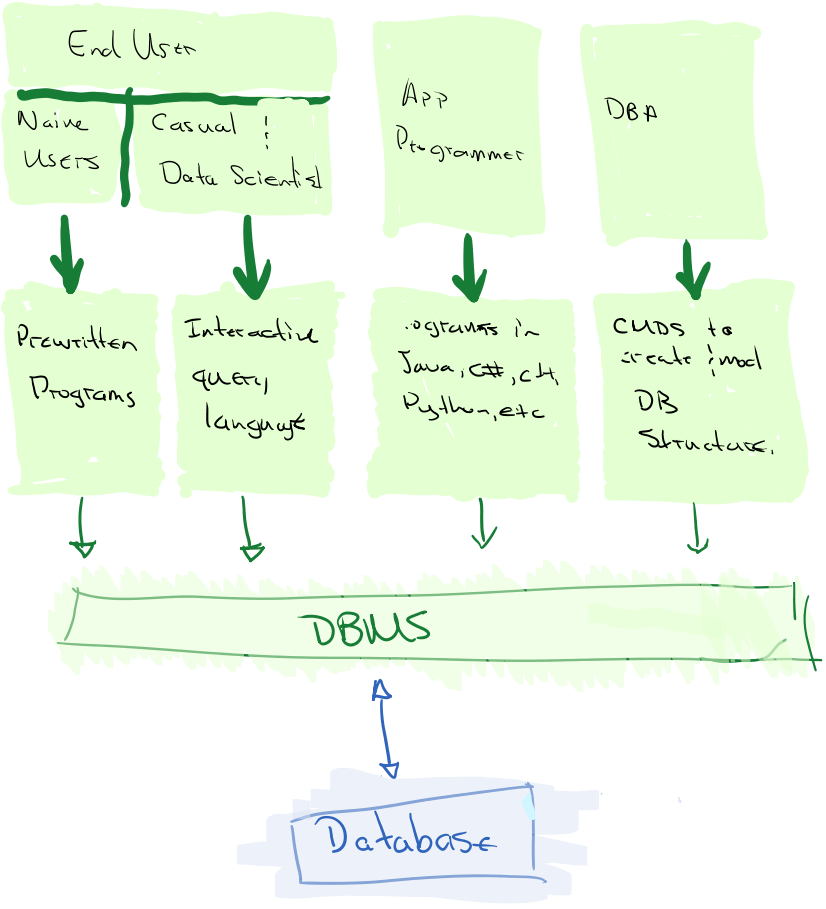
Secondary User
- use DB info w/o interacting with DB

End Users — **Naive**
- do not use query languages
- access data via apps
- simple CMD use

Data Scientist
- ANalyze large amounts of Data w/ query languages & other tools
- Find data trends via tech/AI

Application Programmers
writers of Apps in host prog lang
Java, C, C#, Python, R

Database Administrator
design, create, secure, maintain } Database



Advantages of the Intergrated DB Approach

Advantages of the Intergrated DB Approach

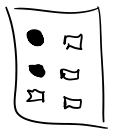
- Sharing of Data
- Economy of Scale
- Consistency
- Balance of conflicting reqs
- Fast dev of new apps

Improved:
- Standards
- Security
- integrity
- accessability
- backup & recovery OP

greater control:
concurrency
redundancy

Historical Developments in IS

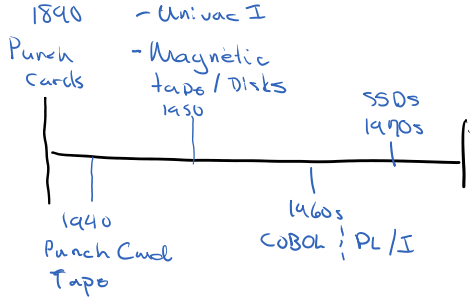
→ Storage Media



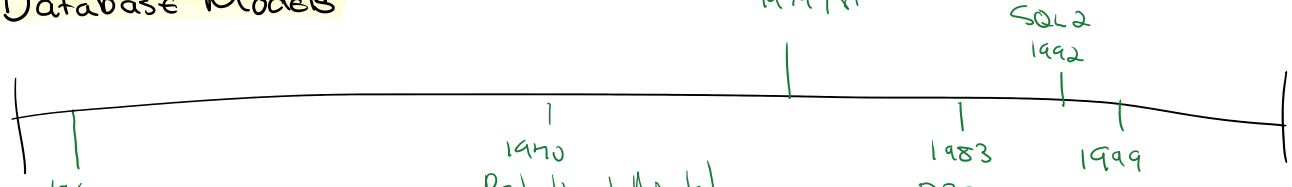
(1890)

- Use of punch cards by US census.
- Herman Hollerith

→ Punch card tape (1940s)
Magnetic tape (1951)
Univac I : first commercially available computer



Database Models



1960s
Hierarchical Model
IBM (IMS)

Intergrated Data Store
Charles Bachman
(GE)

Conference on Data System languages → Data base Task Group

1970
- Relational Model
- E.F. Codd
- System R
- SQL (Structure query language)

Oracle, first Commercial RDBMS - SQL/DS
DB2
1979/81

1983
DB2

1999
SQL3

1976 (Peter Chen)
Entity Relationship (ERI)

2010s
- KML (extended markup language)

1976 (Peter Chen)
Entity Relationship
(ER)
(EER) extended

2010s
Data Lakes

1990s
Object-Oriented
- Unified markup lang (UML)
Grady Booch, I. Jacbson, J. Rumbaugh

Object Relational

Data Warehouses

- KML (extended markup languages)
- JSON (Java Script Object Notation)

Sunday, October 23, 2022 7:21 PM

Chapter 2 DP Planning & Architecture

Stages in DB Design

- **Analyze User Environment** — Study of current practices; inputs & outputs; how system is used
- **Develop a Conceptual Model** — ID of entities, attributes, and relationships
- **Choose a DBMS** — Depends on need; hardware and software available
- **Develop a logical model** — Map conceptual model to the DB model
- **Develop the physical model** — Performance estimates; prototypes
- **Tune System** — Any modifications/updates are made @ this point
- **Implement the physical model** — If eval is good; physical design get green light

All ——— Analyze
Cats ——— Conceptual
Dogs ——— DBMS
Lizards ——— Logical
Paint ——— Physical
Trains ——— Tuning
Pretty ——— Physical implementation

Design Tools

Data dictionary

- directory
- active data dictionary
- integrated dictionary

↓

Archived data that lays out the logical structure of the DB.

Metadata - data about data

- Freestanding Data Dictionaries; not tied to a Database.

Diagramming Tools

Entity relationship :

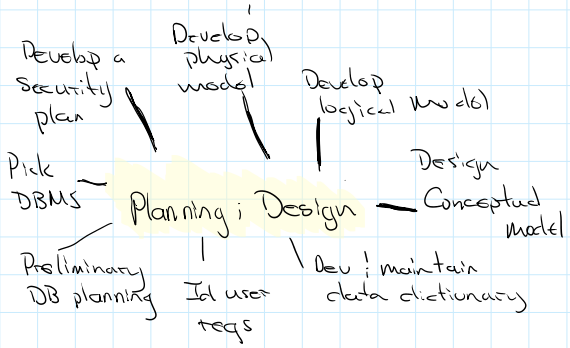
Extended entity relationship :

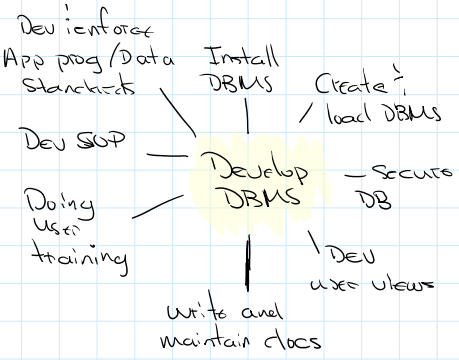
Unified modeling language :

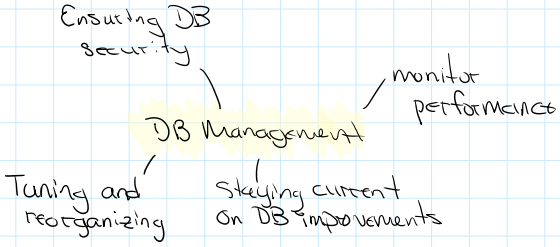
Extended entity relationship:

Unified modeling language:

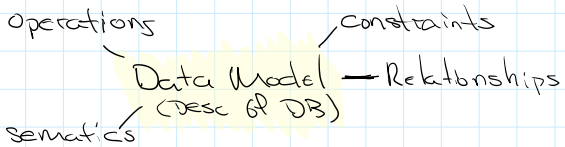
Functions of the DBA







Overview of Data Models



ER Model (Peter Chen, 1970)
- semantic: desc conceptual level of data
- independent of the logical, internal, and physical
- Attributes
  entity characteristics → entity set

Symbol	Name	Meaning
	Rectangle	entity set
	Oval	Attribute
	Diamond	Relationship
	line	links: attibute to entity

Record Base Model
- Draws out logical records
- Network; hierarchal models
  - Tree diagram but nodes can have more than one parent
  - tree with one parent nodes

Object Oriented Model

centered around a obj → person, place, event or concept in the real world.
- state: values of attributes
- behavior: set of operations

Class = entity set
Encapsulation:
OR Model: structured/array of value

Data Warehouse Models

SemiStructured Data Models

Data Warehouse Models

Support of analytical queries
Star schema: large static data support
- dimension table: storing additional info
Datacubes / Hypercubes: multidimensional ~~arrays~~
- pivoting, rollup, drill-down

Semi-Structured Data Models

XML extensible mark-up language
- data exchange over presentation
id elements, sub elements: attributes

HTML hypertext markup language

DTD document type definition
id elements, attributes, relationships to one another

JSON JavaScript Object Notation
- human readable text to rep data as attribute value - pairs

Sunday, October 23, 2022 11:06 PM

Keys

- data item to tell records apart

Superkey

- attribute(s)
- uniquely IDs an entity → Student (StuId, name, major, etc)
  - Student: entity
  - StuId: Superkey

Candidate key

- does not contain extra attributes

  ex → {SocSecNo}

- composite key: more than one attribute

Primary key

- main key to access entities / records
  - alt keys: unique values allow other ways of accessing values.
- Secondary key: means of accessing records (not unique)

ex: Student(StuID, SocSecNo, LastName)

- StuID: PK
- SocSecNo: AK
- LastName: SK

Relationships

- links entities
- connections / interactions

Relationship type: common properties of certain relationships → relationship set
collection of relationships of that type.

Instances: meet the reqs for the membership in the relationship set

Attributes of Relationship Sets

descriptive: belongs to relationship not entity

Cardinality

of enity instances to another entity can map under the relationship.

- One-to-One
- One-to-Many
- Many-to-one
- Many-to-many

One: 1 → —

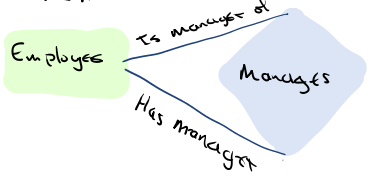
Many: M, N →» — →

Degree of Relationship

of entity sets linked

- Unary
  linked to itself w/ degree of one.

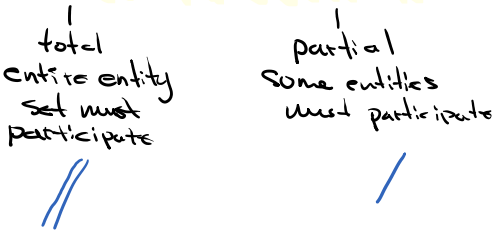
Recursive: entity set related to itself



Participation Constraints:

- total
- partial

Participation Constraints:



Tuesday, October 25, 2022 10:42 AM

The Relational Model

Advantages of the relational Model

- logical
- generalized results
- data independance
- Simple structure
- Seperate conceptual & physical levels
- Mapping is straightforward
- Operations are easily expressed
- few powerful commands for data manipulation.

Relational Data Structures

Tables = Relation
↓
2-D Array

- hold information about objs in DB
- Domain: set of allowable values for that attribute.

Tuple: Rows in ER Table

StuId	lastName	FirstName	Major	Credits
~	~	~	~	~
~	~	~	~	~

Column Name = attributes

Tuples

Student Table → Relation

Table Shows Characteristics:

- Each cell contains one value
- Column, distinct name (attribute being repped)
- Values in Column are from same domain
- No duplicate tuples
- Order of rows does not matter

DB Relations & Tables

- Relation schema (R) set of attributes w/ their corresponding domains.

Set $\{A_1, A_2, \ldots A_n\}$ w/ domains $\{D_1, D_2 \ldots D_n\}$

Relation (r) on a relation schema R is a set of mappings from the attribute names to their corresponding domains.

Mathematical Relations

For two sets (table) A & B → Cartesian product (
A × B: concatenating all A rows w/ B rows.

7 student row × 9 enroll rows
= 63

- Relation = subset of the Cartesian Product

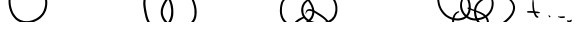
$$\prod_{i=1}^{n} D_i$$

- Instance/extension: table w/ rows written ou
- Intension: relational schema

Student(stuId, lastName, firstName, major
(stuId ↑ PK)

Degree & Cardinality

of columns in table → degree



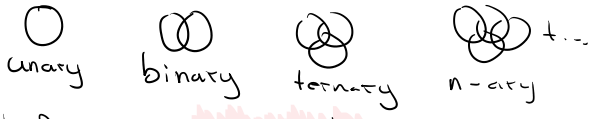
Keys

Superkey : tied to attributes uniquely IDs entities.
Primary key : sk IDs multiple tuples (Rows)

x · join)

)

# of columns in table → degree



# of rows → cardinality

Superkey : tied to attributes; uniquely IDs entities.

Primary key : SK IDs multiple tuples (Rows)

Candidate key : SK w/ no subsets

Foreign key : attribute/combo of a relation that is a PK of some relation (table)

Faculty (FacID, name, subject, department)

Class (classno, facid, room, schedule)

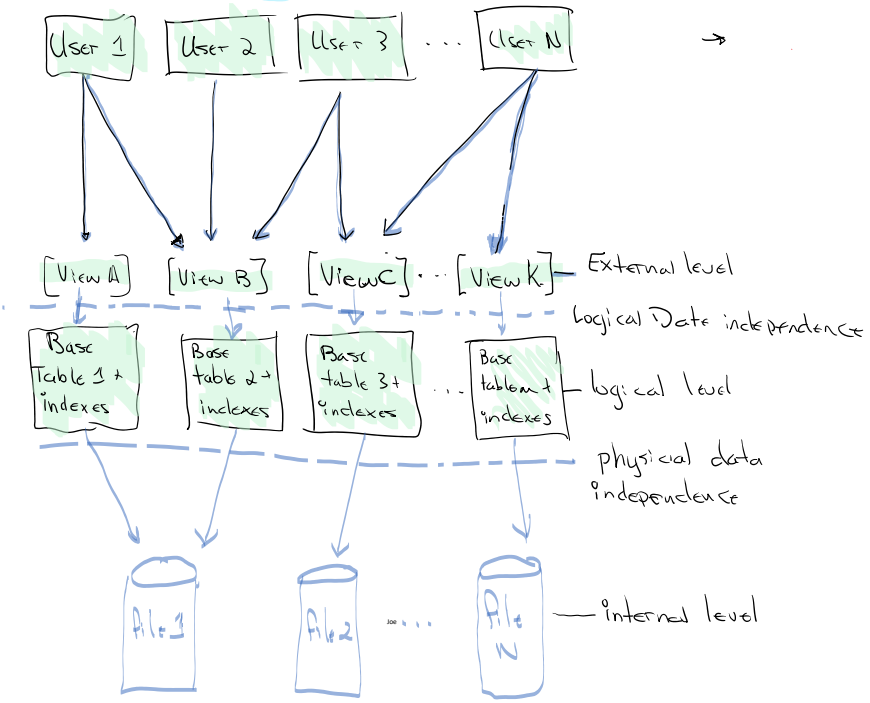
↑ FK

Mapping an Entity-Relationship (ER) Model to a Relational Schema

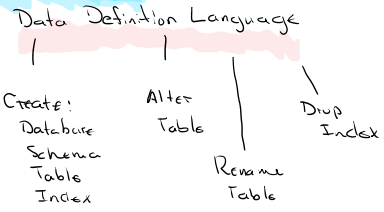
Wednesday, October 19, 2022 11:51 AM

Relational DBMS and SQL

Architecture of a relational DBMS (RDBMS)



Defining the DB: SQL DDL




```
CREATE DATABASE dbname;
    USE dbname;

CREATE SCHEMA schema_name [AUTHORIZATION] username;
CREATE TABLE [schema_name.] base_table_name (colname datatype
[column constraints]
[, colname datatype [column constraints]]
...
[table constraints]
[storage specifications]);
```

★ Composite is a combo of multiple columns/keys

Data types

- Oracle
  - Varchar(2)
    - Varying len str
    - up to 4,000 bytes
  - char(n)
    - fixed len str
    - up to 2000 bytes
  - number(p,s)
    - fixed-point #'s
    - p: precision (max # of significant digits)
    - s: scale (# of digits right of decimal point)

Column & Table Constraints

↓

- appear on the same line as the column
- Table constraints: once all columns or table have been set.

Drop Statements

↓

```
Drop Table basetablename;
```

- Appear on the same line as the column
- Table Constraints: once all columns or table have been set.

CREATE [UNIQUE] INDEX indexname ON basetablename (colname [order] [,colname [order]]...);
ALTER TABLE Basetablename ADD columnname datatype constraints;
Altertable class Add cTitle Varchar2(20);
Altertable basetablename Drop Column columnnname;
Altertable basetablename Modify Column colname [new specification];
Rename table old-table-name To new-table-name;
Alter table . . rename column . . . To . . .;

Drop Statements
↓
Drop Table basetablename;
Drop Index indexname;

Manipulating the DB: SQL DML

Declarative or Nonprocedural:
- allows us to declare what info needs to be fetched w/o telling how to fetch.

→ SQL DML Statements
Select
Update
Insert
Delete

Select

Select [distinct] colname [As newname], [,colname]...
From tablename [alias] [,tablename]...
[where predicate]
[group by group_by_clause] [order by order_by_clause];

Aggregate functions

- count
- Sum
- Avg
- max
- min
- Variance
- Stddev ↑ Standard deviation

"count(*)" is used to list all rows of table

"count" returns only non-null values

- literals: characters that are searched for
- metacharacters that specify what search algorithms are to be used.

Select attlist
From tablename
Where Regexp_like (attribute, regular expression);

Group By: compiles all the records w/ a single value for the unique col.

Select w/ Pattern String

% : Any order, any character, any length >=0
_ (underscore) : Any single character
a...z, A..Z, 0..9, etc. → All other characters stand for themselves.
* Use "Like" to display a pattern of string for character columns
"Not like"
Regexp_like

Operators for Updating

Update: change values in records w/i table

Update tablename
Set columnname = expression
[, columnname = expression]...
[where predicate];

Insert: add records to table

Insert
Into tablename ([colname [,colname]...)]
Values (value [,value]...)

Delete: erases records

Delete
From tablename
where predicate;

Creating & Using Views

Creating & Using Views

Create view viewname
[(viewcolname [, viewcolname]...)] As
Select colname [, colname]...
From basetablename [, basetablename]...
Where condition;

Create materialized view view_name AS subquery;
Modify view → Create or replace view viewname as subquery