Database Administration:

The Complete Guide to Practices and Procedures

Chapter 3
Database Design

Agenda

- From Logical Model to Physical Database
- Database Performance Design
- Denormalization
- Views
- Data Definition Language
- Temporal Data Support
- Questions

Physical Database Design Requirements

- In-depth knowledge of the database objects supported by the DBMS and the physical structures and files required to support those objects
- Details regarding the manner in which the DBMS supports indexing, referential integrity, constraints, data types, and other features that augment the functionality of database objects
- Detailed knowledge of new and obsolete features for particular versions or releases of the DBMS
- Knowledge of the DBMS configuration parameters that are in place
- Data definition language (DDL) skills to translate the physical design into actual database objects

Basic Physical ROTs

- Avoid using default settings
  - They are rarely the best setting
  - It is better to know and explicitly state the actual setting you desire in each case
- Synchronize the logical and physical models
  - Always map changes in one to the other
- Performance before aesthetics
- Meaning: prefer fully normalized
- but deviate when necessary to
  - achieve performance goals
- Almost never say *always* or *never*

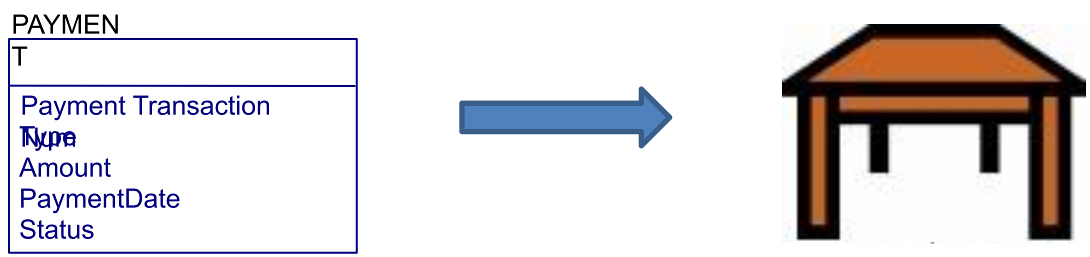
Transforming Logical to Physical

- Translation of Logical Model to Physical Database
  - Create DDL
  - Entities to Tables, Attributes to Columns, Relationships and Keys to DB2 RI and Indexes, etc.
  - …but differences CAN and WILL occur
- Create Storage Structures for Database
  - Files for data and indexes
  - Partitioning
  - Clustering
  - Placement
  - Order of columns

http://datatechnologytoday.wordpress.com/2011/11/19/an-introduction-to-database-design-from-logical-to-physical/

Transform Entities to Tables

- First general step:
  - Map each entity in the logical data model to a table in the database
- Things may, or may not be that easy
  - Denormalization?



Transform Attributes to Columns

- Attributes become columns
- Transform Domains to Data Types
  - Commercial DBMSs do not support domains
  - Date Type and Length
    - Variable or Fixed Length
    - Choose wisely; impacts data quality
  - Constraints
  - Null

tp://craigsmullins.com/dbta_072.htm

Data Types

- CHAR / VARCHAR
- CLOB
- DBCLOB
- BLOB
- GRAPHIC / VARGRAPHIC
- DATE
- TIME
- DATETIME / TIMESTAMP
- XML

- BIGINT
- INTEGER
- SMALLINT
- MONEY
- BINARY
- DECIMAL
- FLOAT
  - REAL
  - DOUBLE

Nulls

INVEN_LOC_TAB

WAREHSE_NO SMALLINT	BIN_NO SMALLINT	PROD_NO CHAR(5)	PROD_QTY INT
1	1	A100	???
1	2	A150	2000
1	3	B167	30
2	1	A100	775
2	2	D400	1585

NULL:

Has no value
Is not = anything
Is not < anything
Is not > anything
Is not = NULL
Is an UNKNOWN value

ATTRIBUTE QUALIFIER	DESCRIPTION
DEFAULT NULL	When no value is provided, DB2 automatically assigns nulls to the Column
NOT NULL	The column must always contain a value, whether a default or explicity provided

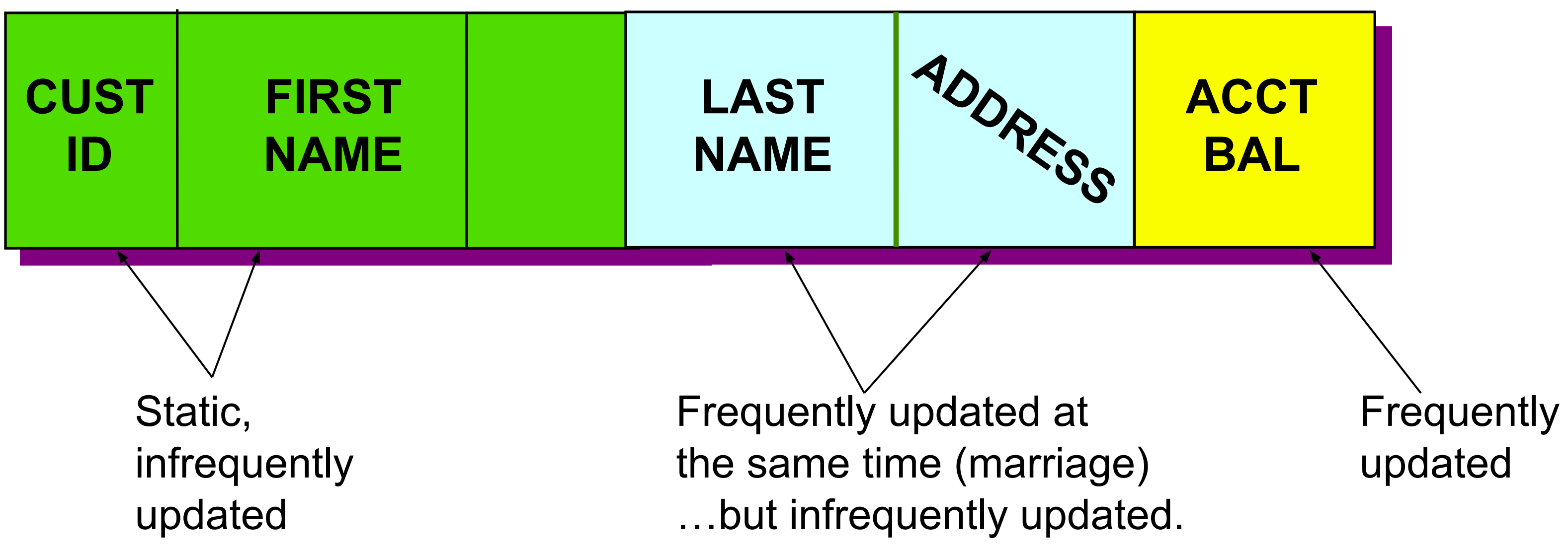
tp://craigsmullins.com/dbta_043.htm

DEFAULT

ATTRIBUTE QUALIFIER	DESCRIPTION	DEFAULT USED
(WITH) DEFAULT	When no value is provided, DB2 automatically assigns an appropriate default	Numeric Data: ZEROS Character Data: SPACES Variable Data: ZERO LENGTH Chronological Data: CURRENT DATE CURRENT TIME CURRENT TIMESTAMP
(WITH) DEFAULT value		Constant USER CURRENT SQLID NULL

Column Ordering

- Sequence columns based on logging. For example:
  - Infrequently updated non-variable columns first
  - Static (infrequently updated) variable columns
  - Frequently updated columns last
  - Frequently modified together, place next to each other



Relationships and Keys

- Use the primary key as assigned in the logical modeling phase for the physical PK
- Other considerations:
  - Length of the key
  - Surrogate key
  - ROWID / SEQUENCE / Identity
- Build referential constraints for all relationships
  - Foreign keys

Build Physical Structures

- Table Spaces
- DBSpaces
- Data Spaces
- Filegroups

Determine Row Size

TABLE Name: ____________________

Column Name	Data Type	Length (in bytes)	Variable (add 2)	Null (add 1)	TOTAL

Data Length:	
Row Overhead:	+8
Physical Row Length:	

Data Lengths:

INTEGER	4	FLOAT (DBL PRECISION)	8
DECIMAL PACKED FORMAT		DATE	4
SMALLINT	2	TIME	3
FLOAT (REAL)	4	TIMESTAMP	10

Storage Planning

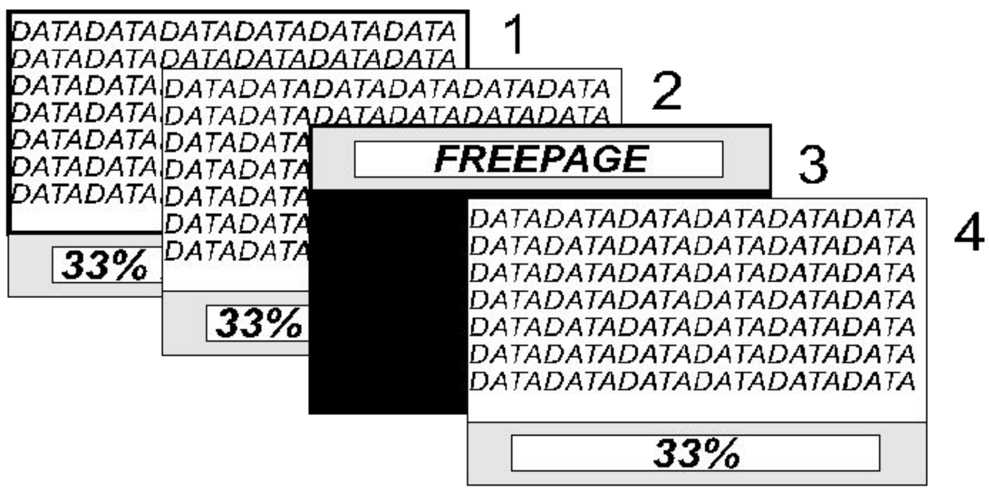
- Start by determining how many rows are required
- Calculate the row size *(discussed earlier)*
- Figure out the number of rows per block/page
- Multiple #rows/page by the page size
- This gives you the size of the object
- Except for free space…

tp://craigsmullins.com/dbta_110.htm

Freespace Alternatives

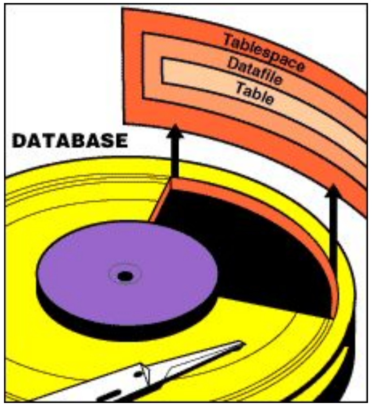
- Freespace is considered only at LOAD or REORG time
- The freespace options:
  PCTFREE
  FREEPAGE
- For Example:

PCTFREE 33 **FREEPAGE 2**



Type of Files

- Data / Index
  - Both require storage
- Raw Files
  - Can be used to bypass the O/S
- Solid State Devices
  - For performance-critical objects
- *Compression*



Database Performance Design

- Designing Indexes
  - Partitioning
  - Clustering
- Hashing
- Interleaving Data

Designing Indexes

Index Advantages

Optimize data access:

- DBMS decides whether or not to use an index
- DBMS maintains all indexes *(modifications incur cost)*
- Table scans can be avoided through index usage
- Recommended on foreign key columns to speed RI access
- Indexes can minimize sorting
- There can be multiple indexes per table to suit the way data is processed
- Create indexes based on workload (not tables)
- If all columns are in the index you can get index-only access (IXO)

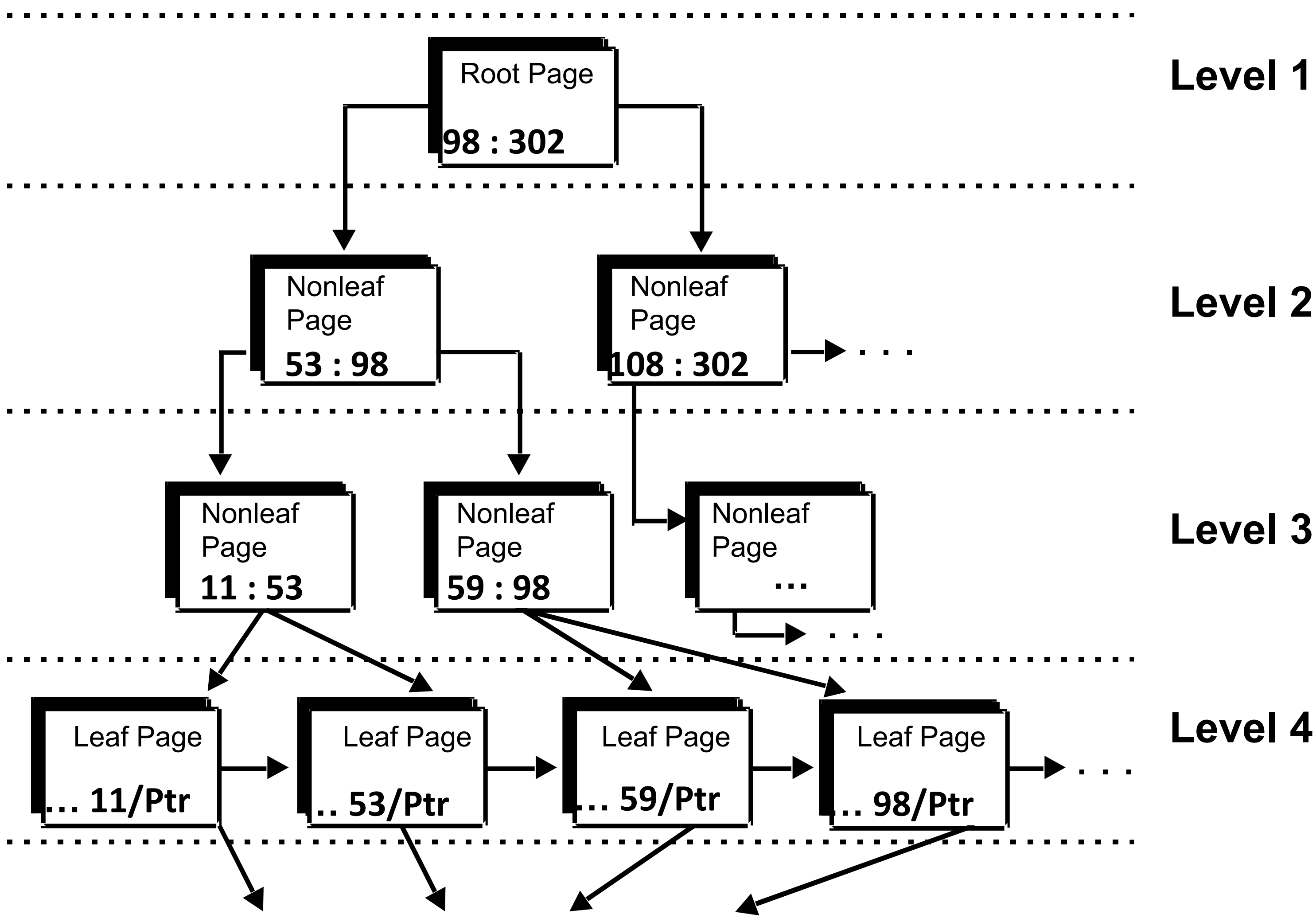
Guarantee uniqueness:

- Can be used to ensure uniqueness of column values
- Required on primary key column as part of referential integrity implementation

Implement clustering:

- Indexes can be used for clustering; that is, maintaining the rows physically on disk in the sequence of the column values in the index

B-Tree Index



…to the data in the table.

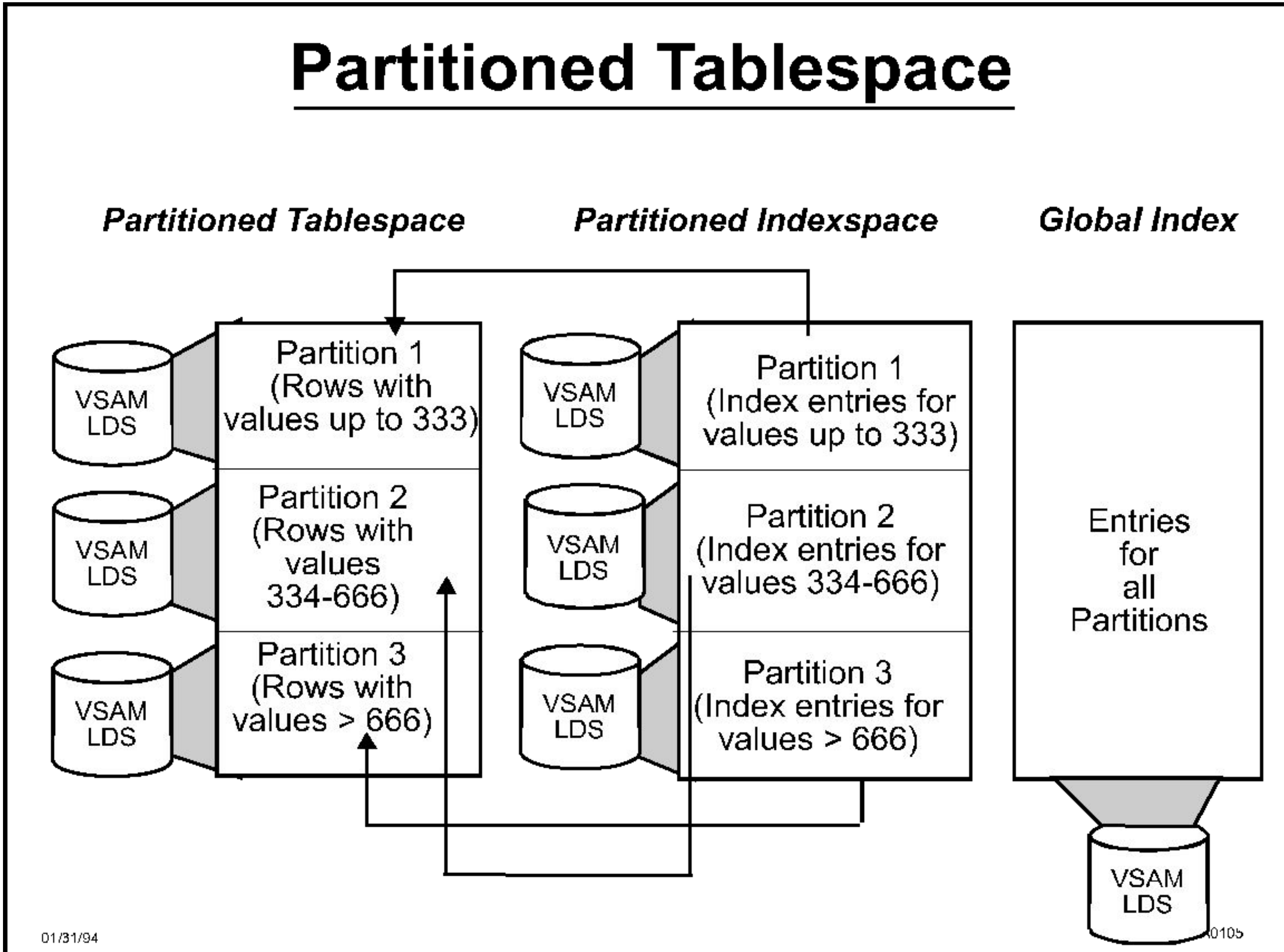
Bitmap Index

Identifier	Gender	Bitmap
1	Female	0110000010
2	Male	1000011101
3	Unknown	0001100000

Other Types of Indexes

- Reverse Key Index
  - a b-tree index where the order of bytes of each indexed column is reversed; helps with hot spots
- Partitioned Index
  - a b-tree index specifying how to break up the index (and perhaps the underlying table) into separate chunks, or partitions; to enhance performance and availability
- Ordered Index

Partitioning



Clustering

Clustering Index

NON-UNIQUE CLUSTERING INDEX ORDR_DATE

SALES_ORDER_TAB

SALES_ ORDER_ NO INT	ORDR_DATE DATE	CUST_NO SMALL INT	SALES_HIST_ CUST_NO SMALL INT	ORDR_AMT DEC(9,2)
1	1997-09-15		2	1923.45
3	1997-09-23		1	2407.53
4	1997-09-23		10	57613.89
5	1997-09-29	3	5	67000.00
6				42345.88
7				122345.61
8				23007.34
9				9823.55
10				223019.27
11				78780.99

ORDR_NO	DATE
000000125	19970923
000004946	19970923
000000013	19971002
000000615	19971002
000008885	19971002
000074155	19971004

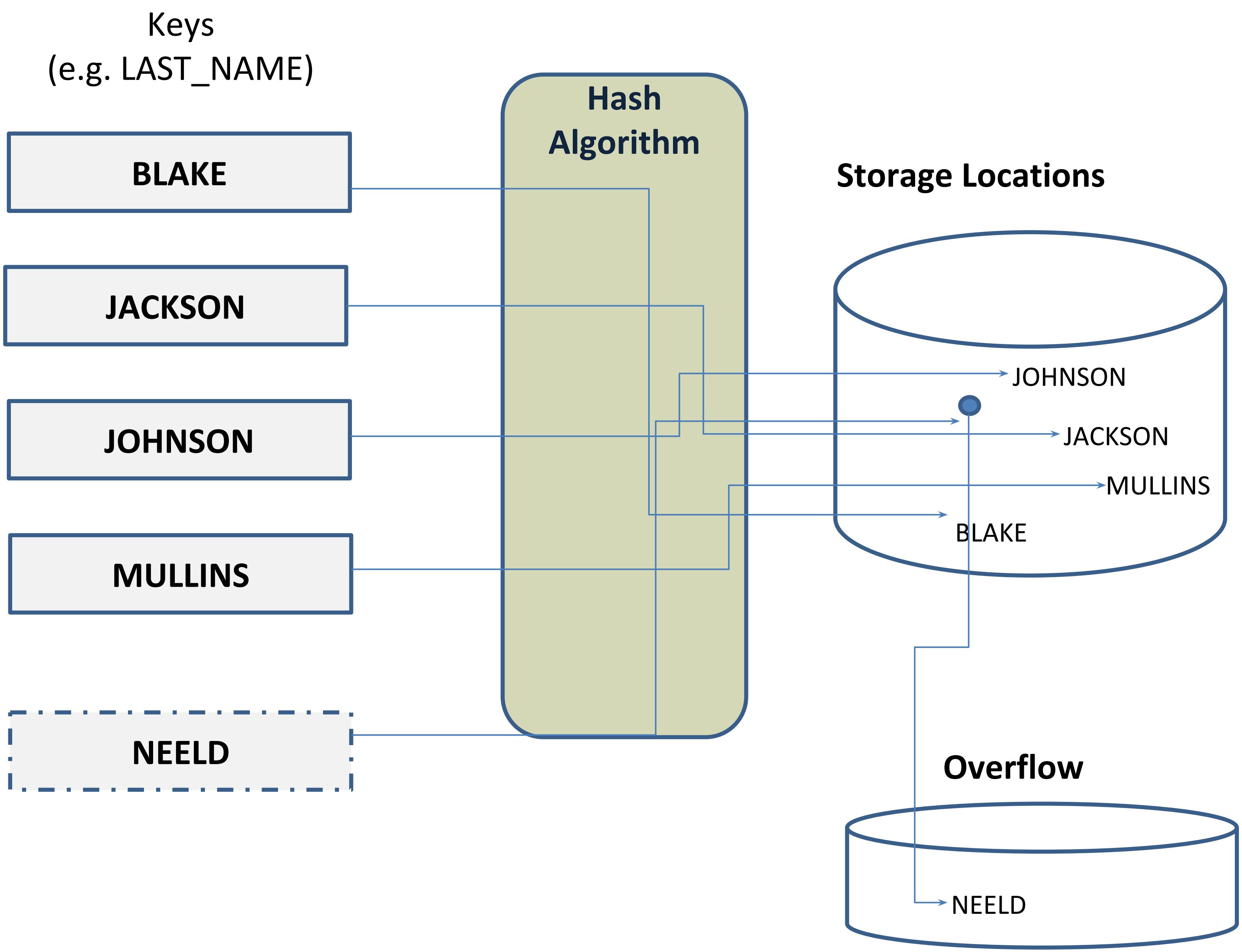
CLUSTERING...

GIVES I/O

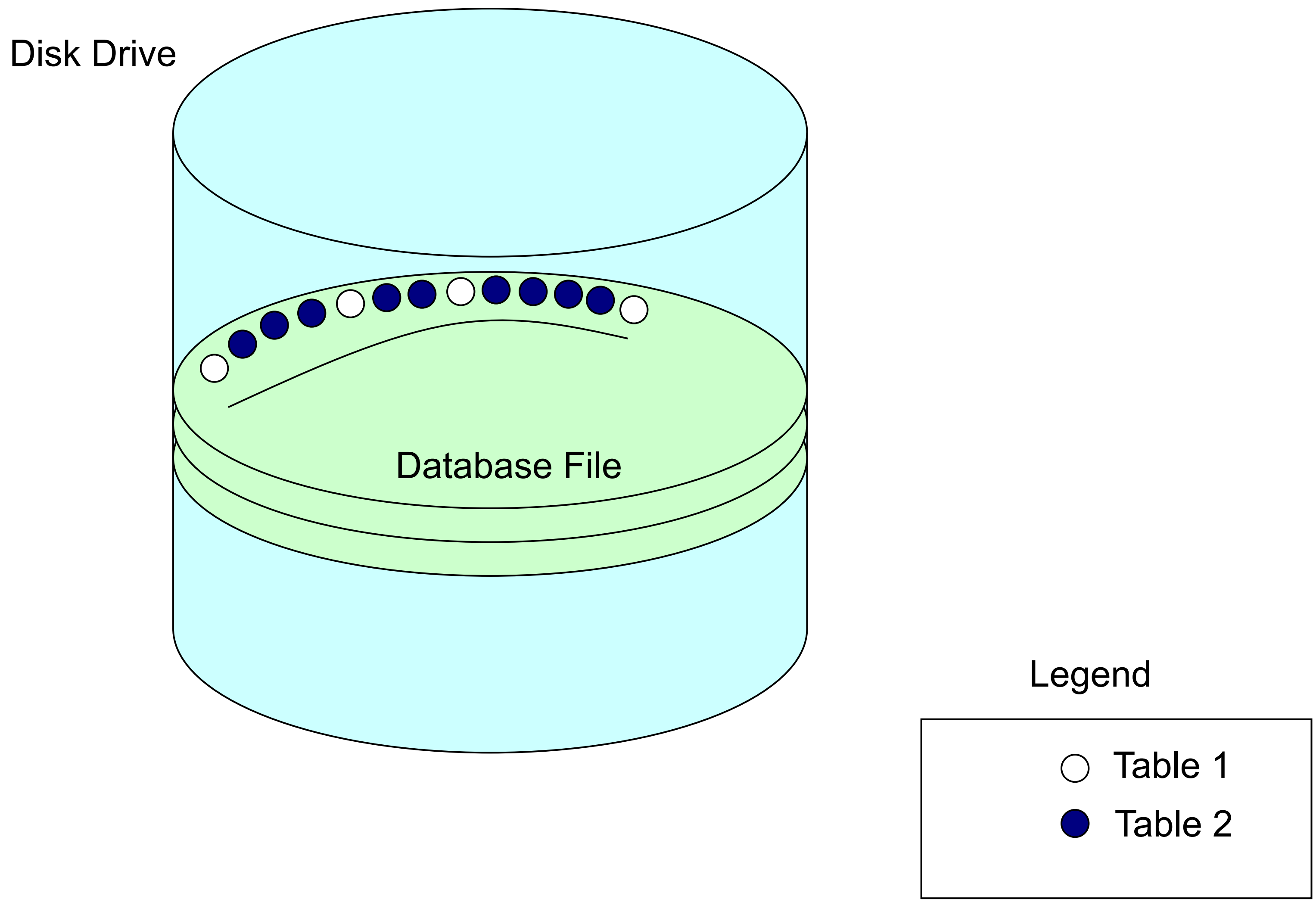
MORE

"BANG FOR THE BUCK!!"

Hashing


Keys
(e.g. LAST_NAME)
BLAKE
JACKSON
JOHNSON
MULLINS
NEELD
Hash Algorithm
Storage Locations
JOHNSON
JACKSON
MULLINS
BLAKE
Overflow
NEELD

Interleaving Data



Denormalization

- **Prejoined Tables** - when the cost of joining is prohibitive
- **Report Tables** - for specialized critical reports (e.g. CEO)
- **Mirror Tables** - when two types of environments require concurrent access to the same data **(OLTP vs DSS)**
- **Split Tables** - when distinct groups/apps use different parts of the same table
  - Splitting **columns** across two tables for long variable character columns.
- **Combined Tables** - to eliminate one-to-one relationships
- **Redundant Data** - to reduce the number of joins for a single column (e.g. definitional, CA to California)
- **Repeating Groups** - to reduce overall I/O (& possibly DASD)
- **Derivable Data** - to eliminate calculations & aggregations
- **Speed Tables** - to support hierarchies
- **Physical Implementation Needs** – e.g.) to reduce page size

tp://www.tdan.com/view-articles/4142

When to Denormalize

The only reason to denormalize, ever:

- To achieve optimal **performance**!
- If the database design achieve satisfactory performance fully normalized, then there is no need to denormalize.

You should always consider the following issues before denormalizing.

- Can the system achieve acceptable performance *without* denormalizing?
- Will the performance of the system *after* denormalizing still be unacceptable?
- Will the system be less reliable due to denormalization?

Denormalization Administration

The decision to denormalize should never be made lightly, because it can cause integrity problems and involve a lot of administration overheads.

Additional administration tasks include:

- Documenting every denormalization decision
- Ensuring that all data remains valid and accurate
- Scheduling data migration and propagation jobs
- Keeping end users informed about the state of the tables
- Analyzing the database periodically to decide whether denormalization is still required

Normalized vs. Denormalized

Normalized Tables:

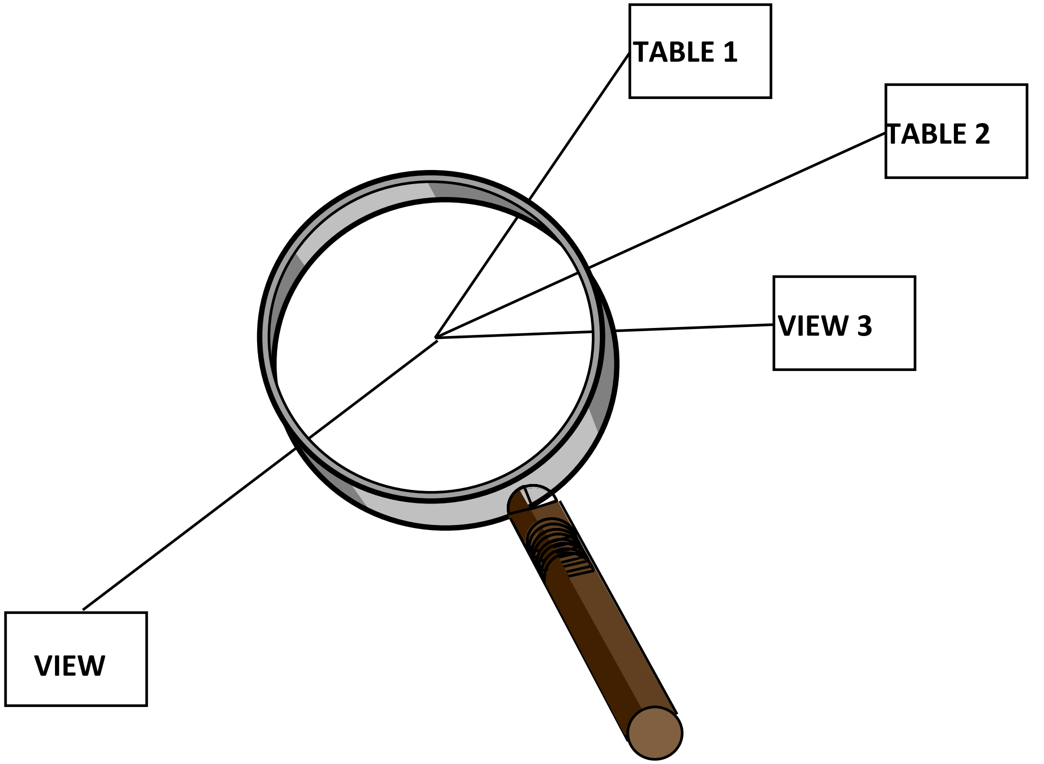
The Goal!

- More Tables
- Fewer Columns per Table
- Fewer Rows per Table
- Less Redundancy
- More Joins
- Update Efficient

Denormalized Tables:

- Fewer Tables
- More Columns per Table
- More Rows per Table
- Greater Redundancy
- Fewer Joins
- Read Efficient

Views



What is a View?

An actual table:

Course Table = TCRSE

CRSE_ NUM INT	CRSE_ CLASS CHAR(6)	CRSE_ INST INT	CRSE_NAME CHAR(20)	CRSE_ DATE DATE	CRSE_ TIME TIME	CRSE_ DAY CHAR(2)	CRSE_ TYPE CHAR(1)
26	DB2004	200	SYSTEMS ADMIN	1990-11-24	10.00.00	TU	A
27	DB2001	--	INTRO TO DB2	1990-11-30	09.00.00	MO	A
28	DB2002	300	SQL PROGRAMMING	1990-11-30	09.00.00	MO	B
29	PHY001	500	RELATIVITY	1990-10-02	09.00.00	WD	A
30	DB2003	--	DATABASE DESIGN	1990-12-07	09.00.00	MO	B
31	DB2001	800	INTRO TO DB2	1990-12-07	09.00.00	MO	A
32	LIT001	600	ENTREPRENEUR	1990-12-09	10.00.00	WD	A
33	DB2001	100	INTRO TO DB2	1990-12-09	09.00.00	WD	B

Course View = VCRSE

CRSE_ NUM INT	CRSE_NAME CHAR(20)
26	SYSTEMS ADMIN
27	INTRO TO DB2
28	SQL PROGRAMMING
29	RELATIVITY
30	DATABASE DESIGN
31	INTRO TO DB2
32	ENTREPRENEUR
	INTRO TO DB2

CRSE_ DATE DATE	CRSE_ TIME TIME	CRSE_ DAY CHAR(2)	CRSE_ TYPE CHAR(1)
1990-11-24	10.00.00	TU	A
1990-1			
1990-1			
1990-1			
1990-1			
1990-1			
1990-1			
1990-12-09	09.00.00	MO	B

A logical view of that table

Course list 'View'

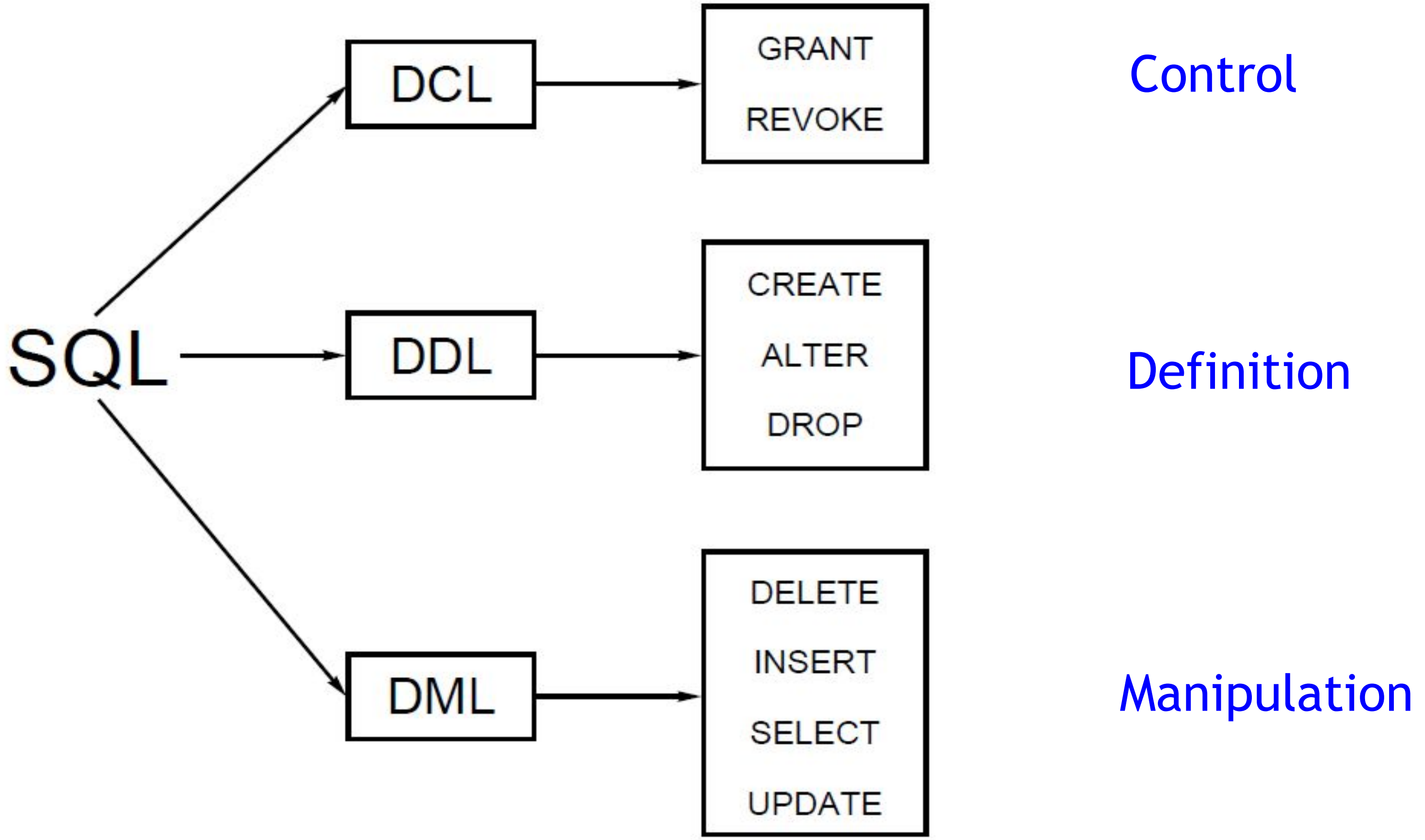
View Usage Rules

- Security - row and column level
- Access - efficient access paths
- Data Derivation - put the calculations in the view
- Mask Complexity - hide complex SQL from users
- Rename a Table
- Column Renaming - table with better column names (easier to use than AS)
- Synchronize all views with base tables...

DO NOT USE ONE VIEW PER BASE TABLE!

tp://craigsmullins.com/dbta_115.htm

Types of SQL



SQL
DCL
DDL
DML
GRANT
REVOKE
CREATE
ALTER
DROP
DELETE
INSERT
SELECT
UPDATE
Control
Definition
Manipulation

Questions