

Object-Oriented Databases

Version Models

- Temporal Databases
- Engineering Databases
- Software Configuration Systems



Overview

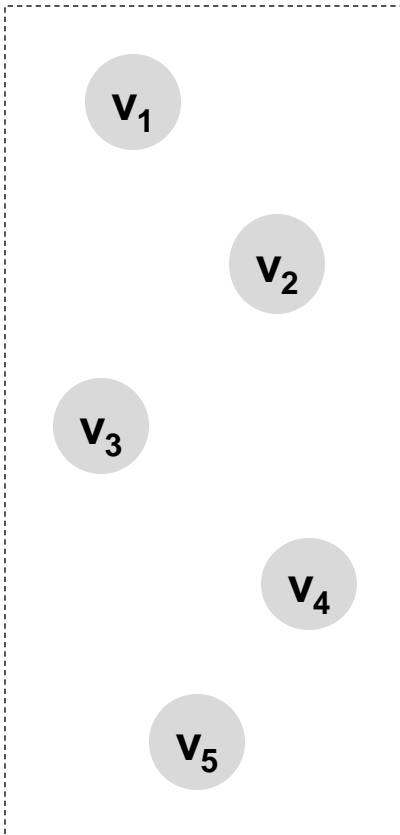
- Various version models have been proposed to meet challenges from application domains
 - temporal databases
 - computer-aided design and computer-aided manufacturing
 - software configuration and software engineering environments
- Evolution of version models
 - very simple approaches at first
 - complex and heterogeneous models emerged
 - several efforts to unify terminology and define generic models
- Association with object-oriented databases
 - version models as motivation for object-oriented databases
 - some object-oriented databases provide versioning support

Versioned Object

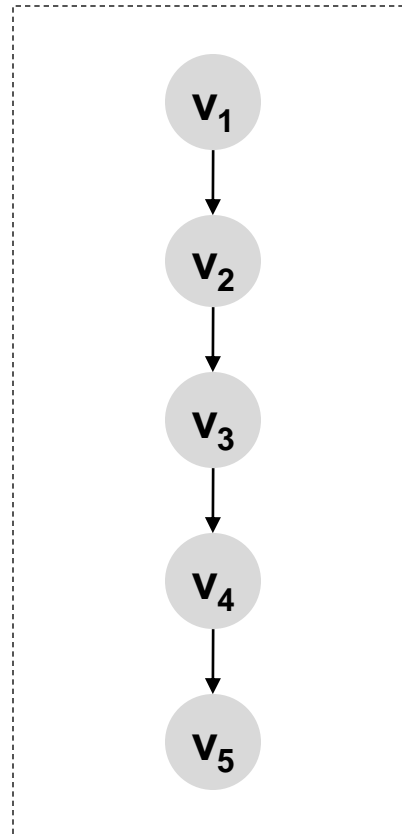
- Concept that has a number of states associated with it
- Different levels of granularity
 - entire files
 - individual tuples of a relation
 - attributes of a class in object-oriented programming
 - objects in object-oriented systems
- Each version is a possible representation of the object, corresponding directly to one of its states
- Interpretation of object states depends on application of version model

Version Organisations

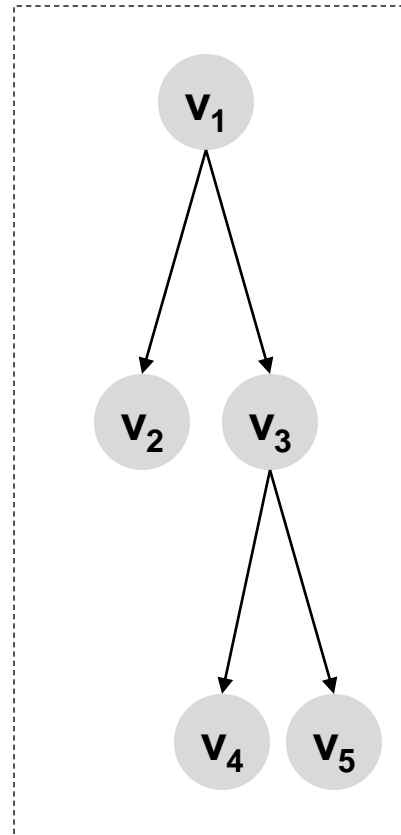
Set



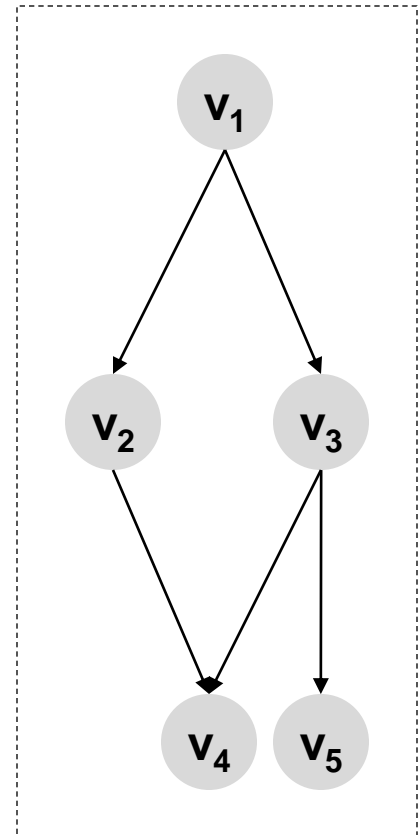
List



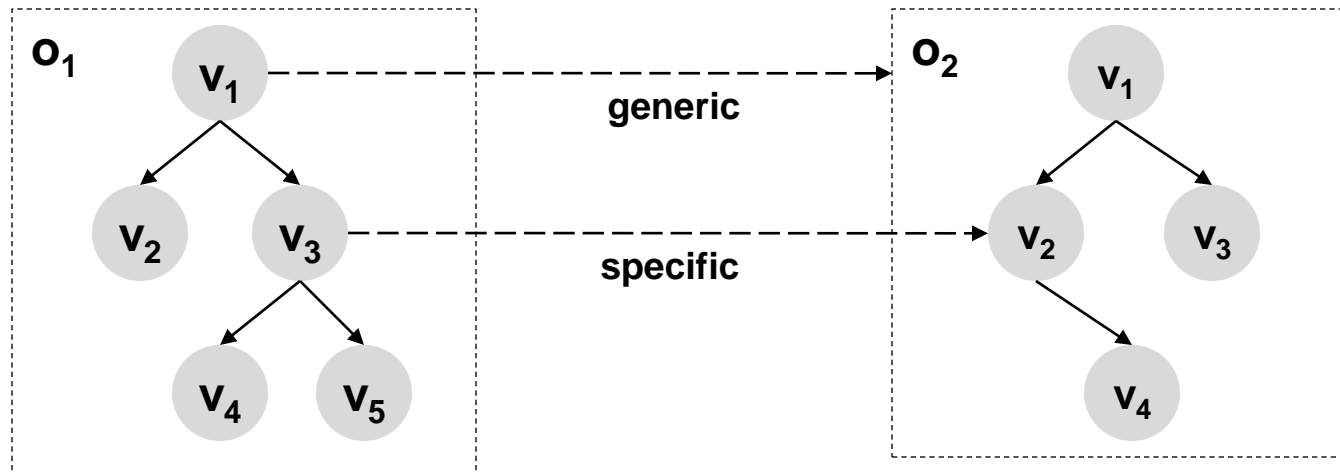
Tree



DAG



References



- Specific reference
 - references single version of object directly
- Generic reference
 - references entire object
 - has to be dereferenced to a version when traversed

Storage Strategies

- Representing versions at the physical level
 - storing complete versions of objects
 - storing changes or deltas between versions
- Delta-based approaches come in several variations
 - forward and backward deltas
 - state-based and operation-based deltas
- Storage and retrieval performance of approaches
 - approaches based on complete versions perform well when changes between versions are substantial
 - delta-based approaches perform well when data changes little, but is not suited to store parallel versions
 - space versus time performance

Operation and Interaction Models

- Operations control evolution of versions of single objects
 - create a new version of an object
 - branch a parallel version of an object
 - merge two parallel versions of an object
 - delete a version of an object
- Interaction or transaction models support working with complex objects and objects graphs
 - automatic versioning is transparent to the user
 - library model uses check-out and check-in high-level operations
 - long running and nested transactions

Queries and Configurations

- Queries over versioned objects involve additional constraints to select correct representations
- Various implementations exist
 - configurator evaluates rules against versioned object network
 - declarative queries express constraints in extended language
 - logical unification based on feature logic
- Dereferencing of generic references
 - query evaluator needs to select specific version of an object
 - main derivation guides generic access for parallel versions
 - active versions guide generic access for sequential versions
 - main derivation and active versions can be used in conjunction to dereference a generic reference

Temporal Databases

- One of the first application domains for version models
- Manage different flavours of time-dependent data
- Vast field of research with numerous approaches
 - conceptual models
 - data models
 - storage models
 - temporal algebras
 - query languages
- Research in temporal databases done mostly based on relational databases systems

Time in Databases

- Different types of time can be used to characterise temporal data
- Transaction, registration or physical time
 - captures when values were stored in the database
 - AS-OF operation
- Valid or logical time
 - used to express when values existed in real world
 - WHEN operation
- User-defined time
 - all aspects of time not covered by other two notions of time

Classification of Temporal Databases

- Static or snapshot database
 - conventional database
 - does not manage temporal data
- Static roll-back database
 - keeps track of transaction time
 - supports AS-OF operation
- Historical database
 - keeps track of valid time
 - supports WHEN operation
- Temporal database
 - keeps track of both transaction and valid time
 - supports both AS-OF and WHEN operation

Representing Temporal Data

Object Versioning

- object is extended with attribute capturing temporal dimension
- can be realised without violating the relational first normal form

Employee	Office	Salary	T _S	T _E
Anne	A 12	5500	2000	now
Bob	B 34	4000	2002	2003
Bob	B 34	5500	2003	now
Charles	C 56	6700	1995	2000
Charles	C 56	7500	2000	2006
Charles	C 56	7000	2006	now
Denise	B 34	3000	1990	1995
Denise	B 34	5300	1995	2002

Attribute Versioning

- each attribute is extended with temporal information
- requires non-first normal form NF² relational systems

Employee	Office	Salary
Anne	A 12	(5500, 2000, now)
Bob	B 34	(4000, 2002, 2003) (5500, 2003, now)
Charles	C 56	(6700, 1995, 2000) (7500, 2000, 2006) (7000, 2006, now)
Denise	B 34	(3000, 1990, 1995) (5300, 1995, 2002)

Conceptual and Data Models

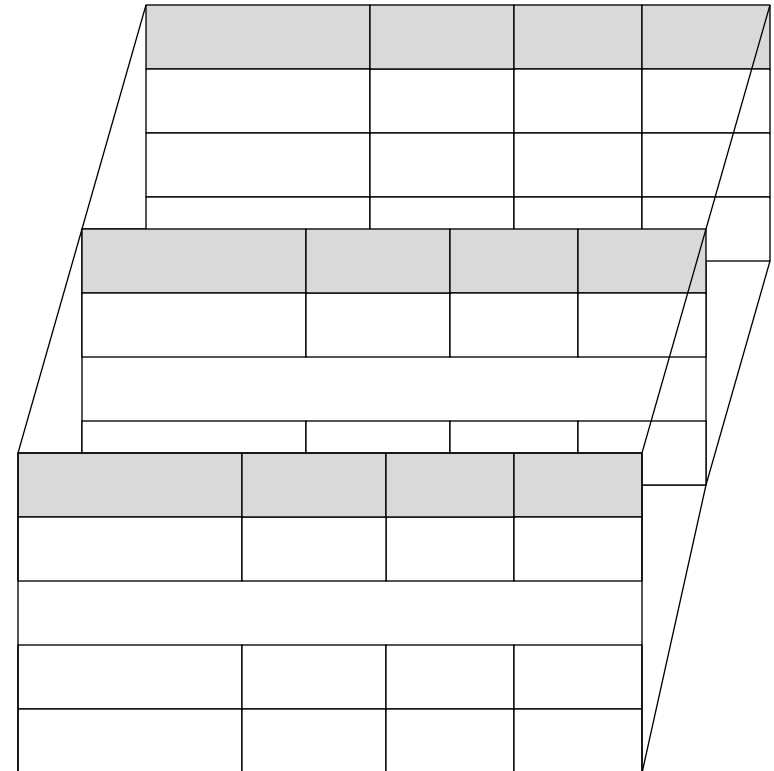
- Early approaches extended existing models such as the relational model or the E/R model
- Bitemporal Conceptual Data Model (BCDM)
 - tuple versioning
 - implemented using four additional columns per tuple
 - transaction time and valid time with special "until changed" and "now" values to indicate if a tuple is current
 - query language TSQL2 is an extension of SQL that introduces a VALIDTIME and WHEN clause
 - TSQL2 has been integrated into SQL3 as SQL/Temporal

Homogeneous and Heterogeneous Models

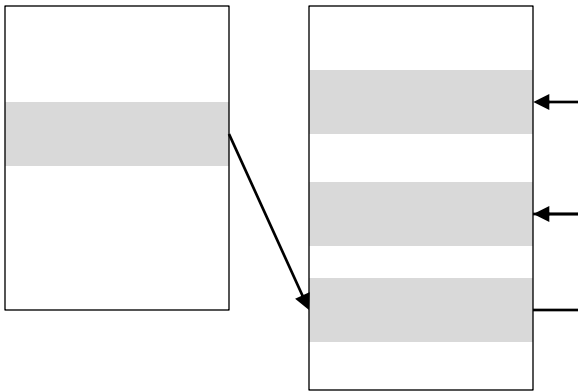
- Temporal data model is homogeneous if the temporal domain does not vary from one attribute of an object to another
- All models that use tuple versioning are homogeneous
- Heterogeneous models can suffer from two anomalies
 - if a horizontal anomaly is present, a versioned object is spread across several records in different data sets
 - if a vertical anomaly is present, a versioned object is spread across several records of the same data set
- Anomalies also apply to object-oriented databases on the physical level

Storage Models

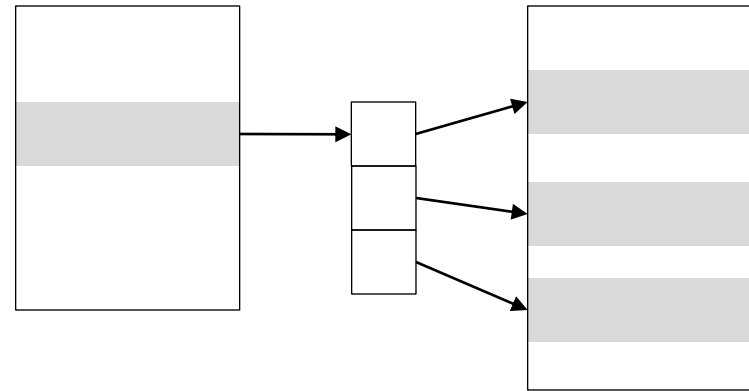
- Temporal relation can be viewed as tree-dimensional data structure
 - sequence of relations
 - data cube
- Implemented using a two level store structures
 - primary store contains current versions which satisfy all non-temporal queries
 - history store hold the remaining history versions
- Traditional access methods cannot be used on such a storage model



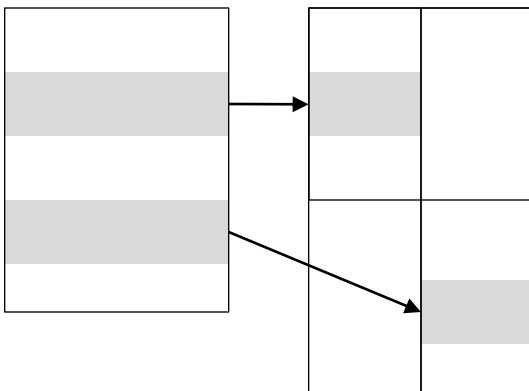
Two-Level Storage Structures



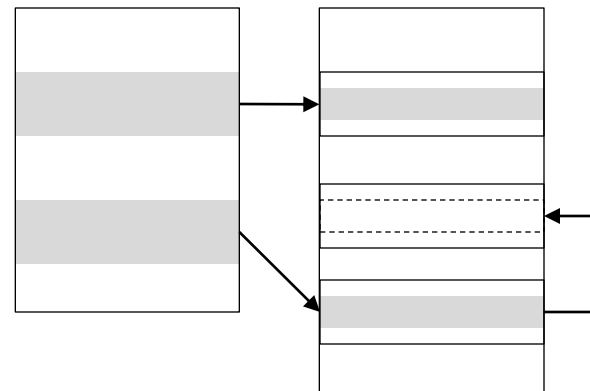
Reverse Chaining



Accession Lists



Clustering



Stacked Versions

Cellular Clustering
(Cell Size $C=3$)

Engineering Databases

- Developed for engineering application domains
 - Computer-Aided Design (CAD)
 - Computer-Aided Manufacturing (CAM)
- Support the development and maintenance of products
- Requirements
 - data structures and concurrency control concepts to define and manage complex, often hierarchical, design objects
 - versioning support for complex objects that supports iterative development by alternatives and trial-and-error experiments
- Two dimensional version models
 - linear revision dimension
 - non-sequential variation dimension

Design Space Version Model

- Modelling primitives
 - component hierarchies (*is-a-part-of*)
 - version histories (*is-a-kind-of*, *is-derived-from*)
 - configurations combine component hierarchies and version histories
 - equivalences capture different representations of a “multi-faceted” design object
- Design management operations
 - identifying the current version within a version history
 - describing dynamic configurations
 - managing the movement of objects among workspaces
 - change and constraint propagation
 - inheriting attributes from related design objects

Early Approaches

- Extension of IBM System R relational database system with long fields and complex objects
- Long fields used to store and retrieve unstructured information of arbitrary length
 - data is written and read using extended cursor concept
 - iteration over stream representing data of a long field
- Complex objects manage several tuples as an object
 - new column types COMP_OF, INDETIFIER and REF introduced
 - component tuples reference other tuples of the same object or root tuple of another object
 - concurrent access based on check-out/check-in model

Software Configuration Systems

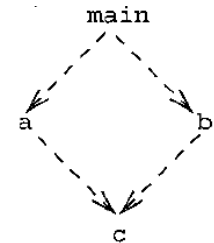
- Developed for software development
 - Software Configuration Management (SCM)
 - Software Engineering Environments (SEE)
- Software configuration systems manage product directly
 - engineering databases only manage a product representation
 - goal of fully automating process of building final product
- Also built around concept of design objects
 - source code files
 - modules of programs
- Management of references and dependencies more complex as hidden inside source code files

Product Space

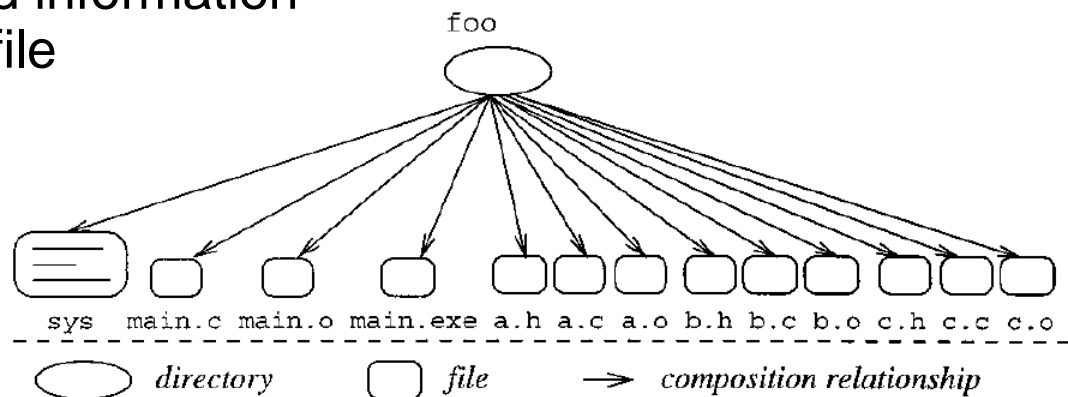
- Describes a product is organised
 - does not take versioning into account
 - several design choices available
 - can be represented in different ways
- Software objects
 - requirements specifications, designs, documentations, program code, test plans, test cases, user manuals, project plans, ...
 - object identification
 - object granularity
 - object representation
- Relationships
 - composition relationships
 - dependency relationships

Product Space Representation

- Logical structure
 - module **main** imports **a** and **b**
 - module **a** and **b** both import **c**
 - product space may be represented in different ways
- File system
 - each module is represented by multiple files
 - dependencies and build information are stored in separate file

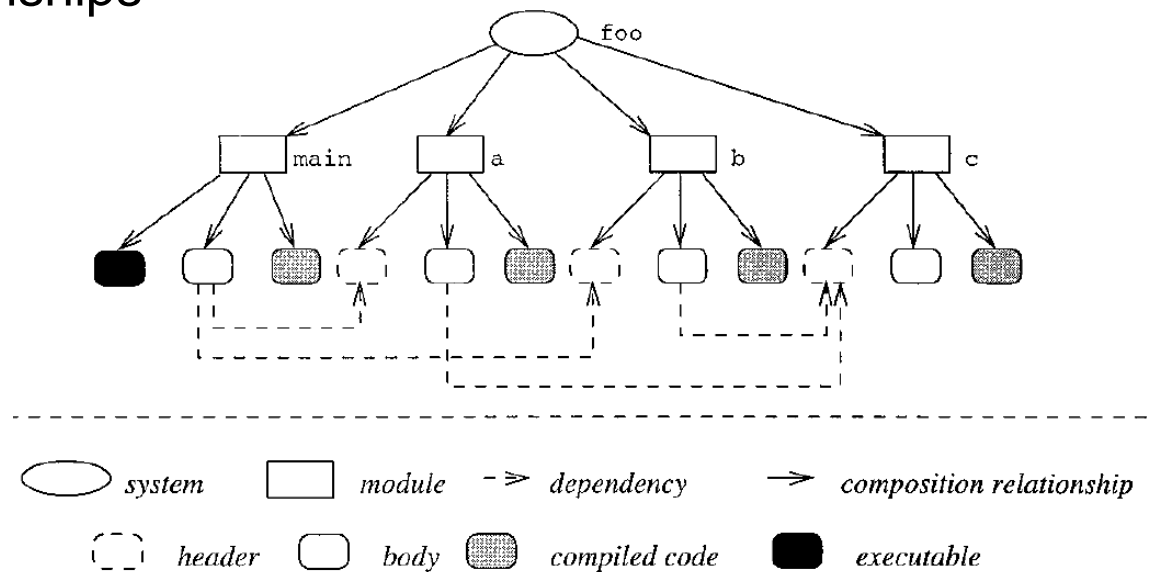


main module
- \Rightarrow import



Product Space Representation

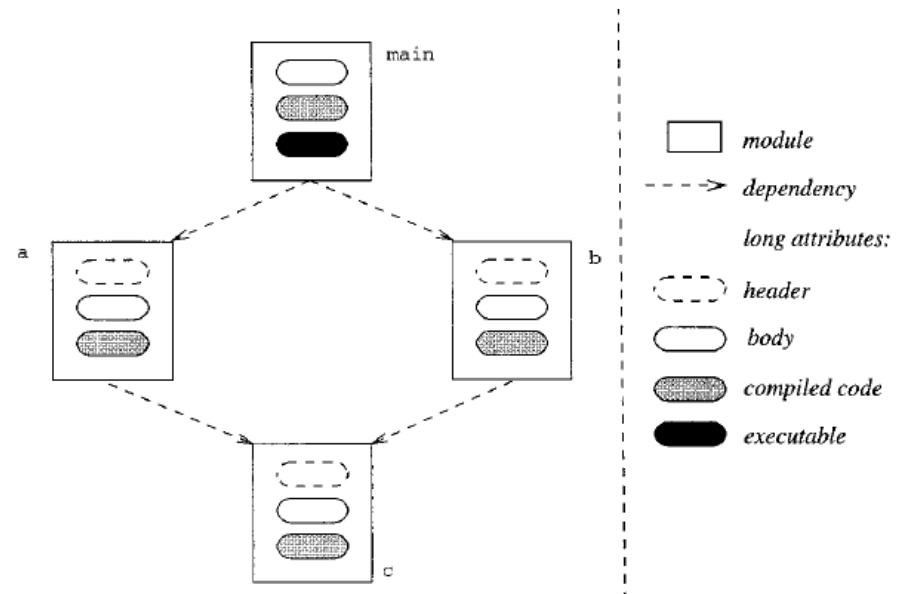
- Data model with type relationships
 - composition tree with files as leaves
 - dependencies are represented within tree
 - build information can be computed from composition and dependency relationships



Reidar Conradi and Bernhard Westfechtel: **Version Models for Software Configuration Management**, In: *ACM Computing Surveys*, 30(2), 232-282, 1998

Product Space Representation

- Without spanning tree
 - all files are summarised as one object
 - only source dependencies are represented
 - directly corresponds to the logical structure



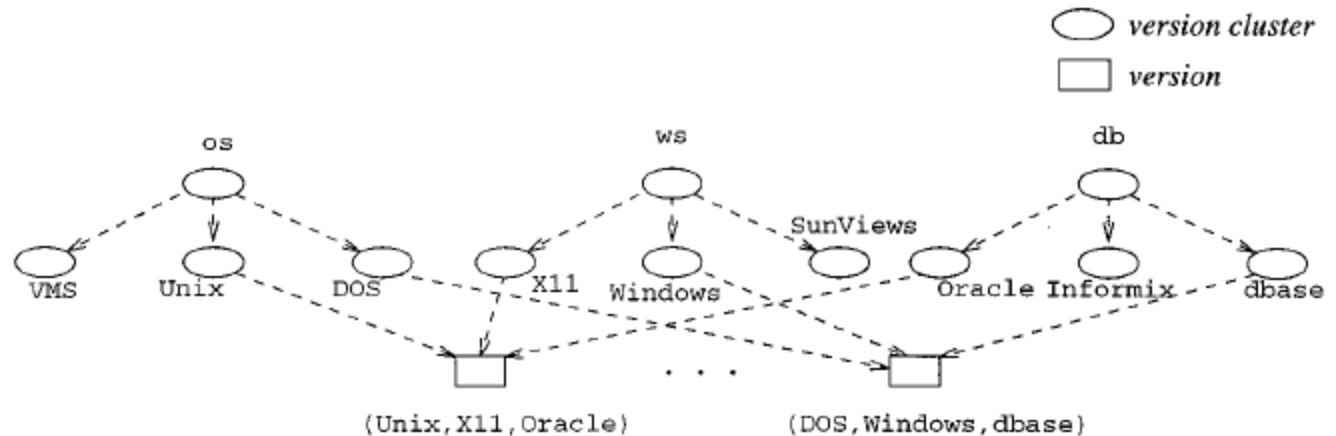
Reidar Conradi and Bernhard Westfechtel: **Version Models for Software Configuration Management**, In: *ACM Computing Surveys*, 30(2), 232-282, 1998

Version Space

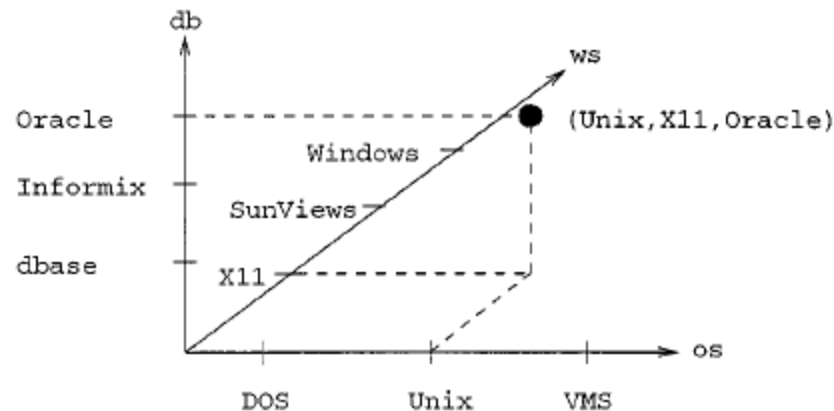
- Version model defines how objects are versioned
 - a versioned object is a container for a set of versions
 - common properties shared by a versioned object (*invariants*)
 - differences (*deltas*) between versions of a versioned object
 - symmetric deltas
 - directed deltas (changes)
- Definition of version set
 - extensional versioning enumerates all members of the version set
 - intensional versioning uses predicate defining version set members
- Intent of evolution
 - revisions keep track of the history
 - variants capture alternatives
 - versions can also be used to support cooperation and collaboration

Version Space Representation

■ Graph



■ Grid



Reidar Conradi and Bernhard Westfechtel: **Version Models for Software Configuration Management**, In: *ACM Computing Surveys*, 30(2), 232-282, 1998

Literature

- Richard T. Snodgrass and Ilsoo Ahn: **A Taxonomy of Time in Databases**, In: *Proceedings of ACM SIGMOD*, 236-246, 1985
- Randy H. Katz: **Toward a Unified Framework for Version Modeling in Engineering Databases**, In: *ACM Computing Surveys*, 22(4), 375-409, 1990
- Reidar Conradi and Bernhard Westfechtel: **Version Models for Software Configuration Management**, In: *ACM Computing Surveys*, 30(2), 232-282, 1998

Next Week

Storage and Indexing

- Type Hierarchy Indexing
- Aggregation Path Indexing
- Collection Operations

