SLEEPYCAT SOFTWARE

…the embedded database company tm

Gizmo Databases

Margo Seltzer
Sleepycat Software and Harvard University

June 10, 1999

What Is a Gizmo Database?

- Gizmo:
  - – A device, not a general-purpose computer
  - – Application oriented
  - – Examples: toaster, telephone, lightswitch
  - – Also: an LDAP server, messaging servers, DHCP servers
- A gizmo database is a database for a gizmo.

Why Do Gizmos Have Databases?

- Gizmos have computers.
- Once there is a computer, people can't help but collect data.

These are not your normal Enterprise databases.

Outline

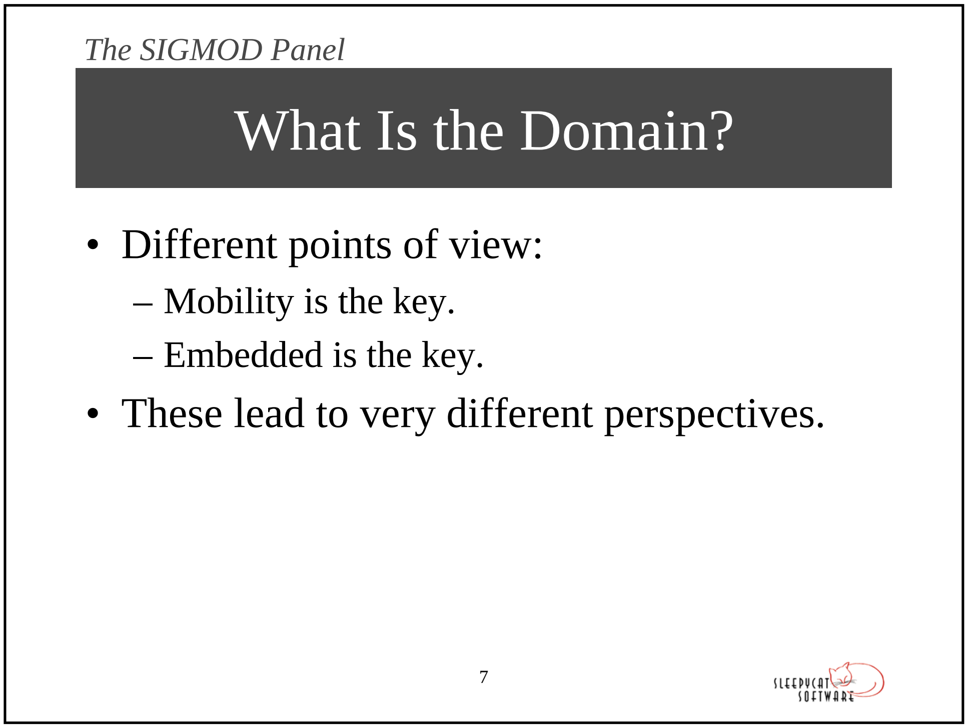
- A summary of the 1999 SIGMOD panel on "small" databases.
- Working definition of an embedded database.
- Challenges in embedded databases.
- Berkeley DB as an embedded database.
- Conclusions.

The SIGMOD Panel on Gizmo Databases

- Honey, I shrunk the database.
  - Emphasis on mobility more than embedded.
- Panelists
  - CTO: Cloudscape
  - VP of mobile and embedded systems: Sybase
  - Founder of Omniscience: built ORDBMS that was sold to Oracle as Oracle Lite
  - Me

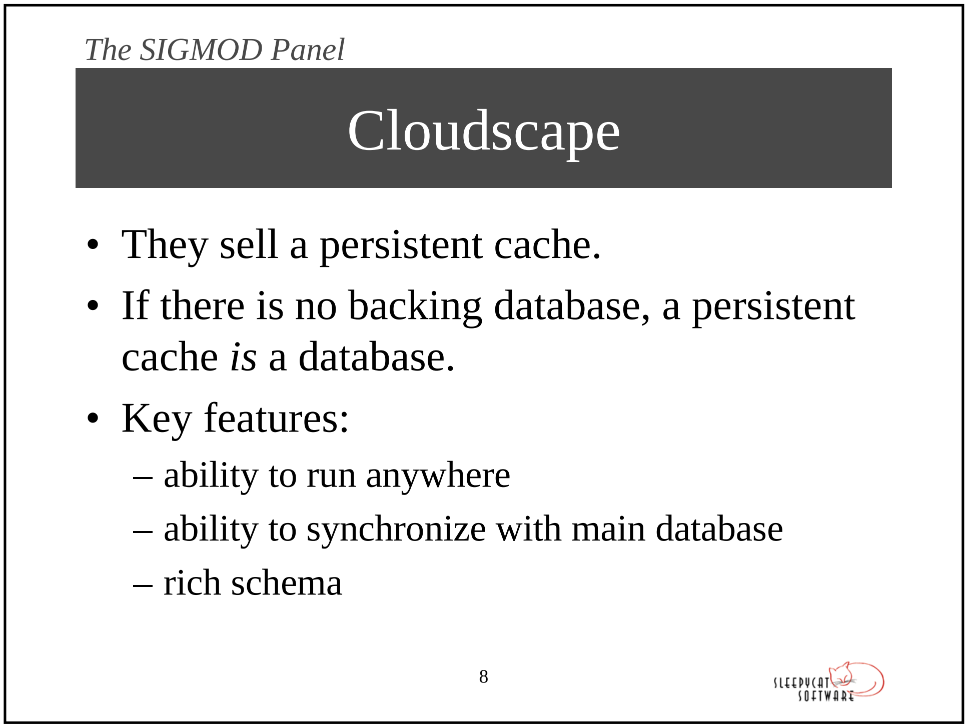
Caveats

- You are getting my (biased) interpretation of the panel.
- You are also getting my (biased) definition of what embedded database systems are.
- You are getting my (biased) definition of what is important.



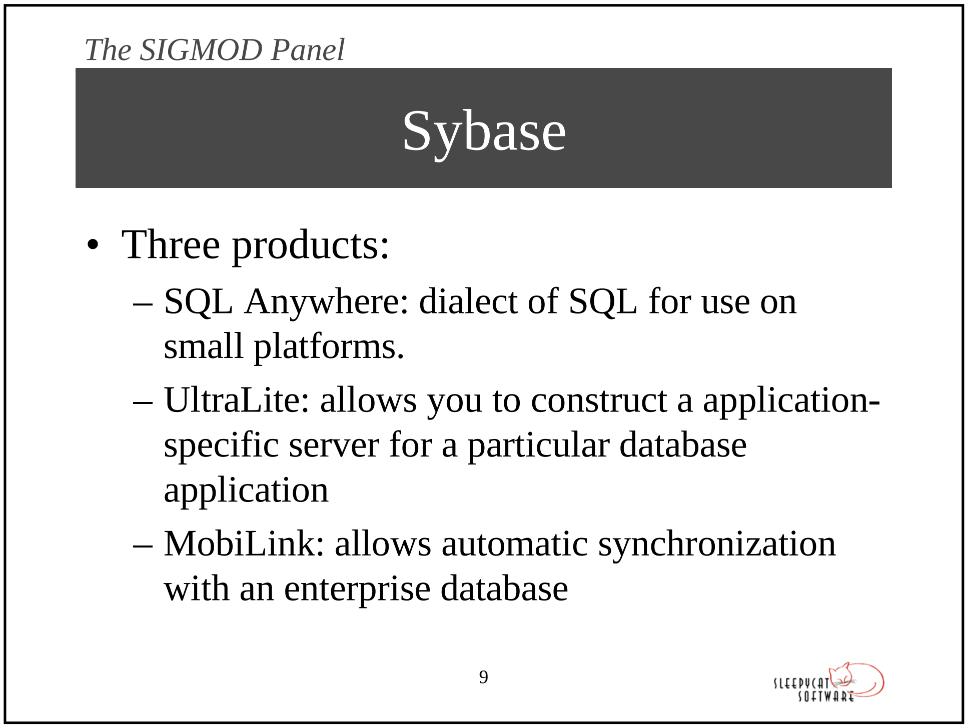
What Is the Domain?

- Different points of view:
  - – Mobility is the key.
  - – Embedded is the key.
- These lead to very different perspectives.



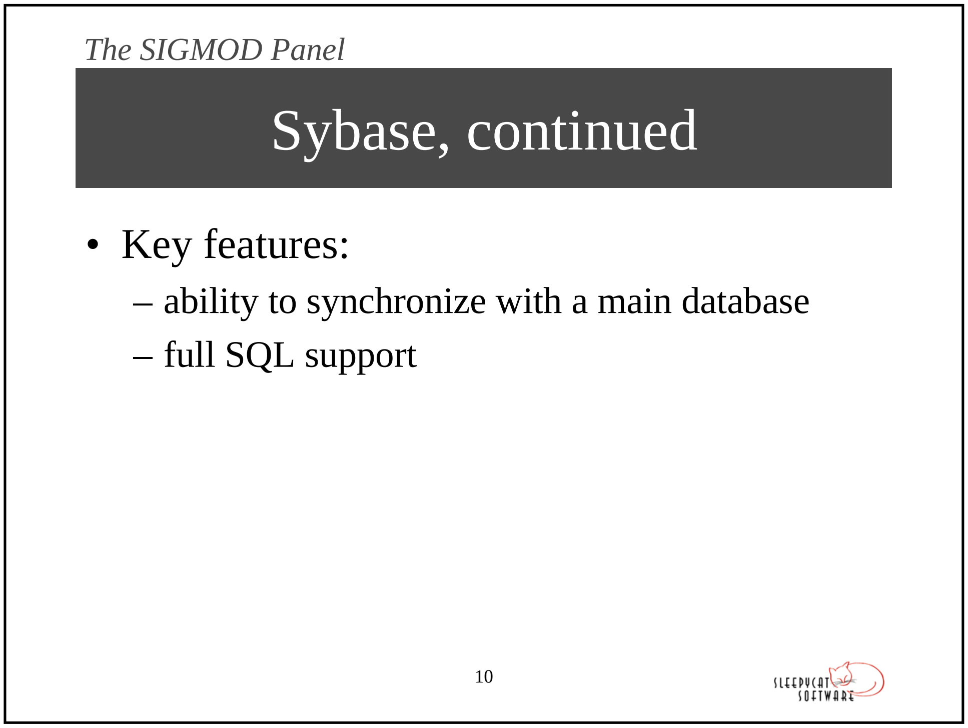
Cloudscape

- They sell a persistent cache.
- If there is no backing database, a persistent cache *is* a database.
- Key features:
  - – ability to run anywhere
  - – ability to synchronize with main database
  - – rich schema



Sybase

- Three products:
  - SQL Anywhere: dialect of SQL for use on small platforms.
  - UltraLite: allows you to construct a application-specific server for a particular database application
  - MobiLink: allows automatic synchronization with an enterprise database



Sybase, continued

- Key features:
  - ability to synchronize with a main database
  - full SQL support

Oracle/Omniscience

- Developed with small footprint in mind.
- (Omniscience) Goal was robustness, not mobile or embedded support.
- Oracle target is mobile applications.

Oracle/Omniscience, continued

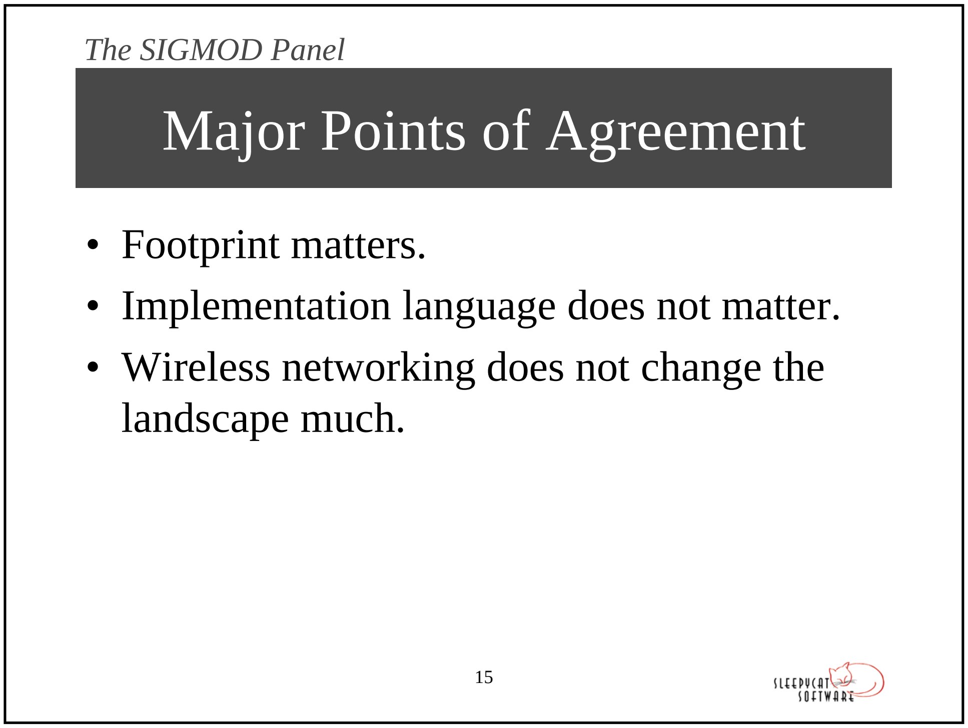
- Key features:
  - small footprint
  - Object-relational model
  - Java support
  - database synchronization

Sleepycat

- Target is embedded applications, not mobile.
- "Users" are other programs, not people.
- General-purpose query interface not important.

Sleepycat, continued

- Key features:
  - transparency (can't tell you exist)
  - small footprint
  - high performance
  - not necessarily related to any enterprise application



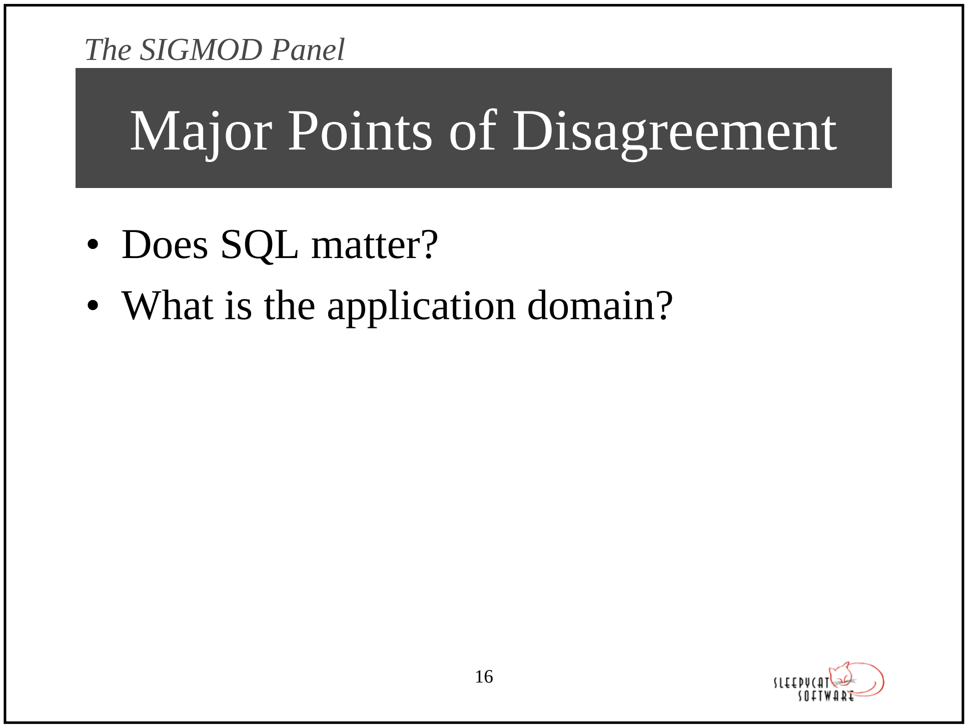
The SIGMOD Panel

Major Points of Agreement

- Footprint matters.
- Implementation language does not matter.
- Wireless networking does not change the landscape much.

15

SLEEPYCAT SOFTWARE



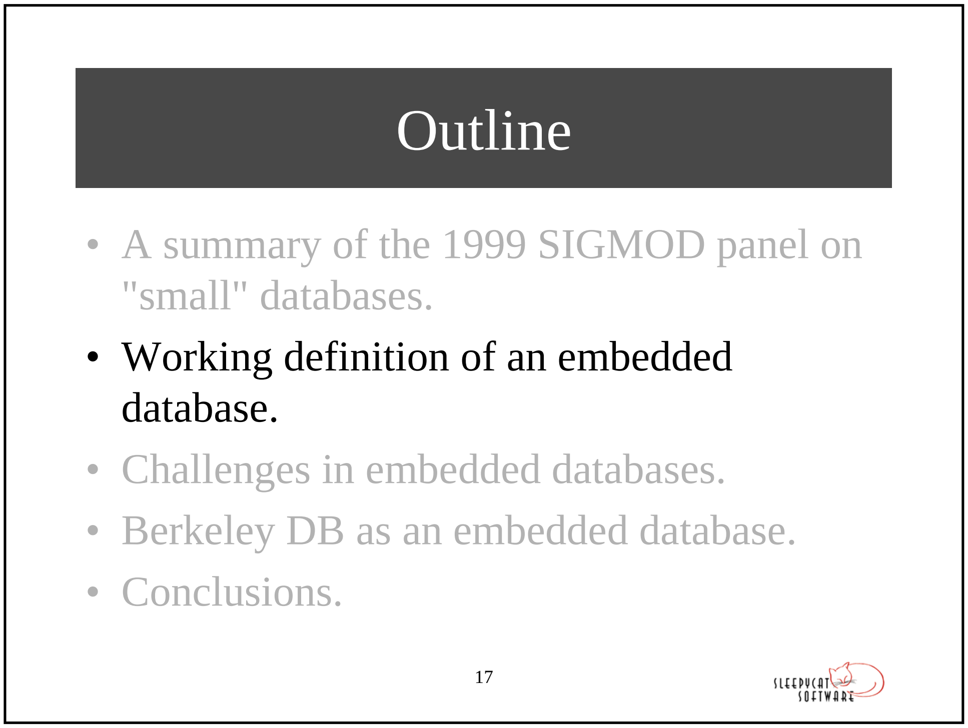
The SIGMOD Panel

Major Points of Disagreement

- Does SQL matter?
- What is the application domain?

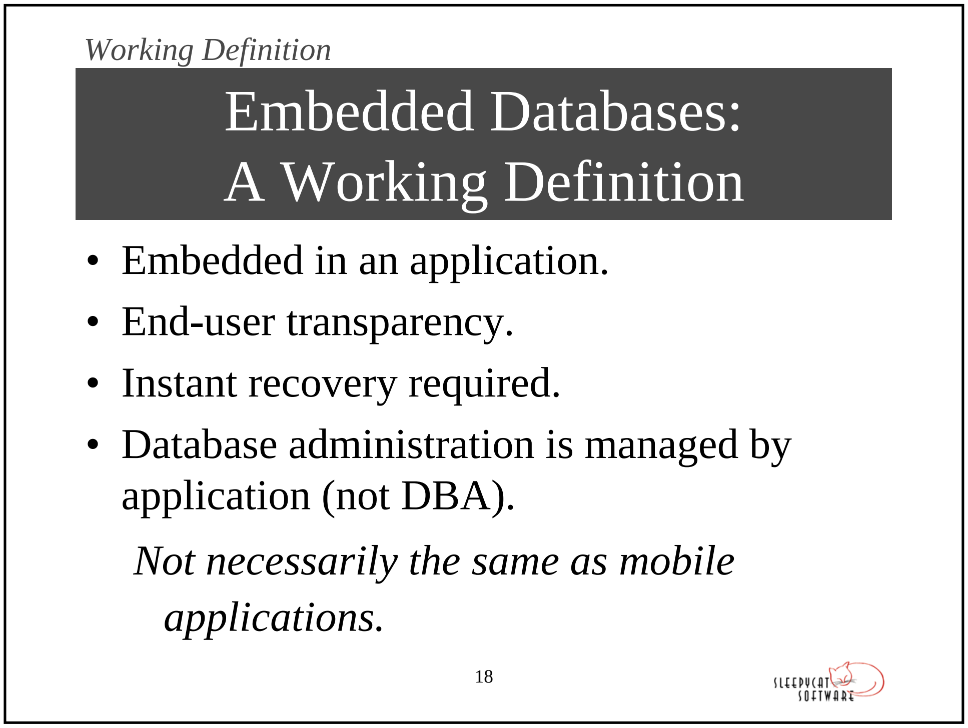
16

SLEEPYCAT SOFTWARE



Outline

- A summary of the 1999 SIGMOD panel on "small" databases.
- Working definition of an embedded database.
- Challenges in embedded databases.
- Berkeley DB as an embedded database.
- Conclusions.

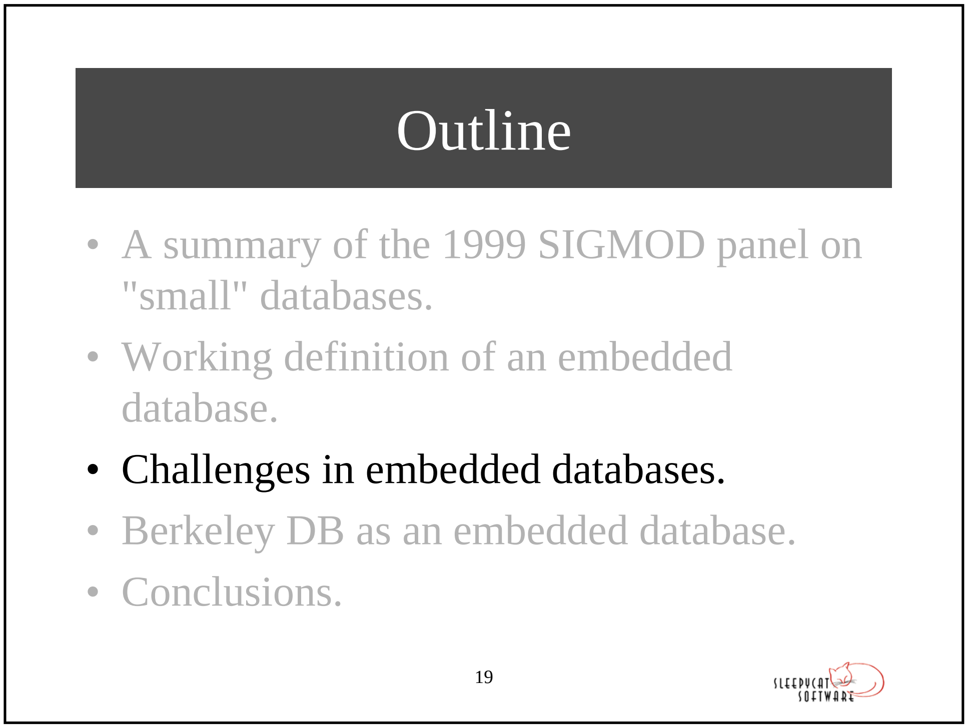


Working Definition

Embedded Databases: A Working Definition

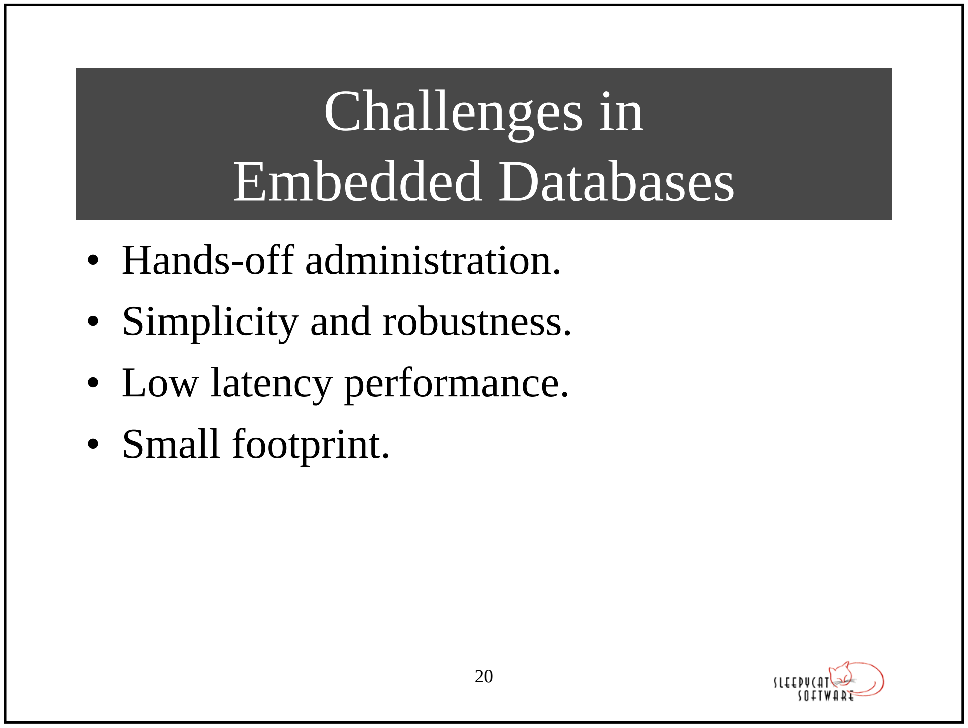
- Embedded in an application.
- End-user transparency.
- Instant recovery required.
- Database administration is managed by application (not DBA).

Not necessarily the same as mobile applications.



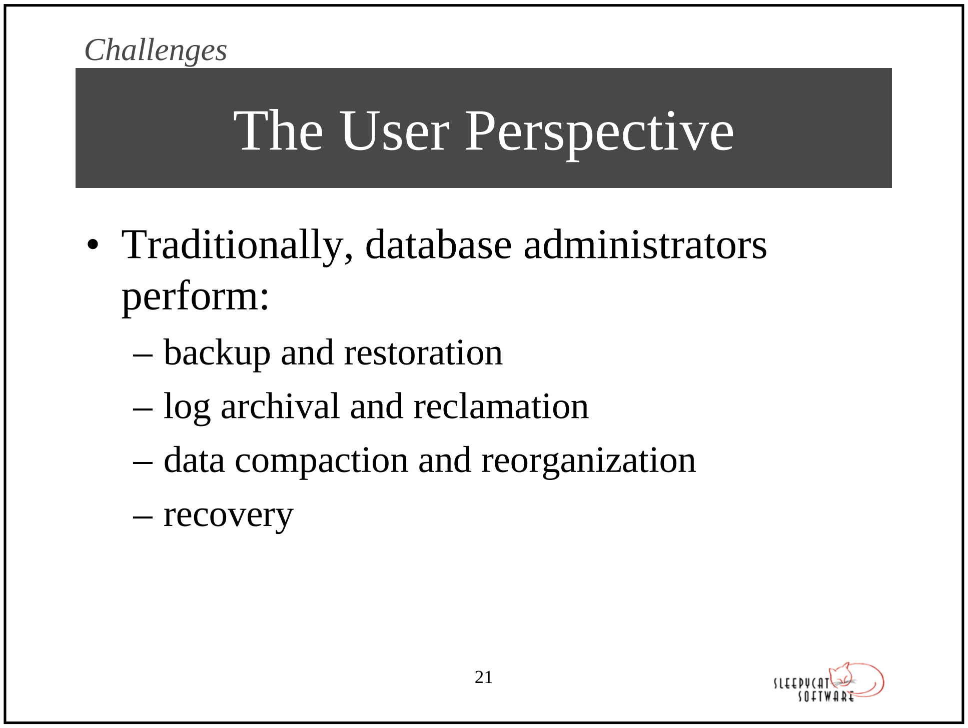
Outline

- A summary of the 1999 SIGMOD panel on "small" databases.
- Working definition of an embedded database.
- Challenges in embedded databases.
- Berkeley DB as an embedded database.
- Conclusions.



Challenges in Embedded Databases

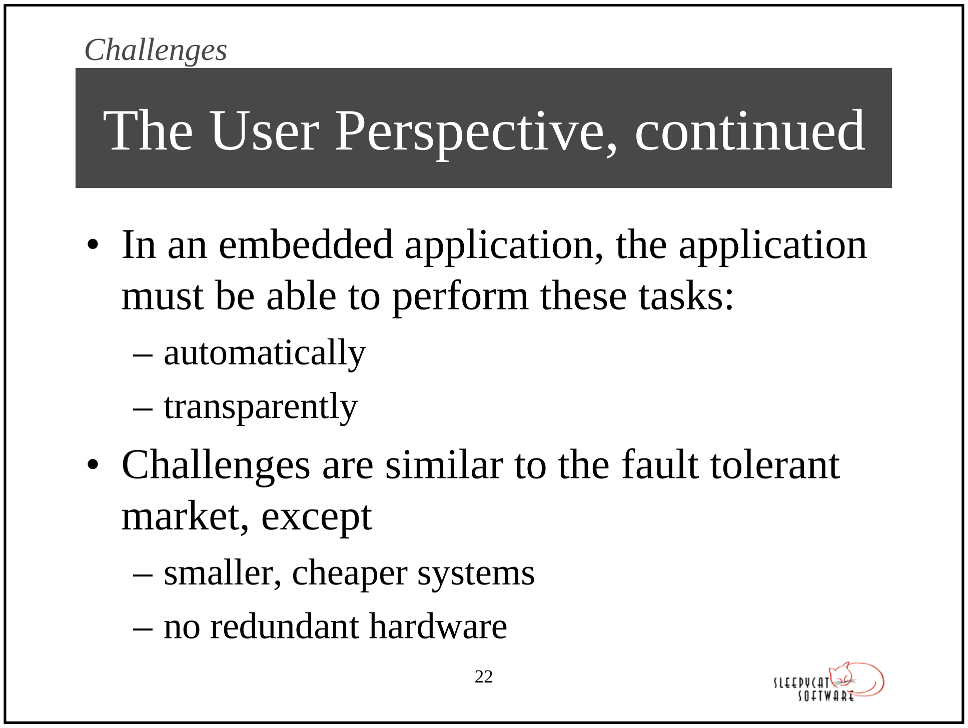
- Hands-off administration.
- Simplicity and robustness.
- Low latency performance.
- Small footprint.



Challenges

The User Perspective

- Traditionally, database administrators perform:
  - backup and restoration
  - log archival and reclamation
  - data compaction and reorganization
  - recovery



Challenges

The User Perspective, continued

- In an embedded application, the application must be able to perform these tasks:
  - automatically
  - transparently
- Challenges are similar to the fault tolerant market, except
  - smaller, cheaper systems
  - no redundant hardware

Challenges: The User Perspective

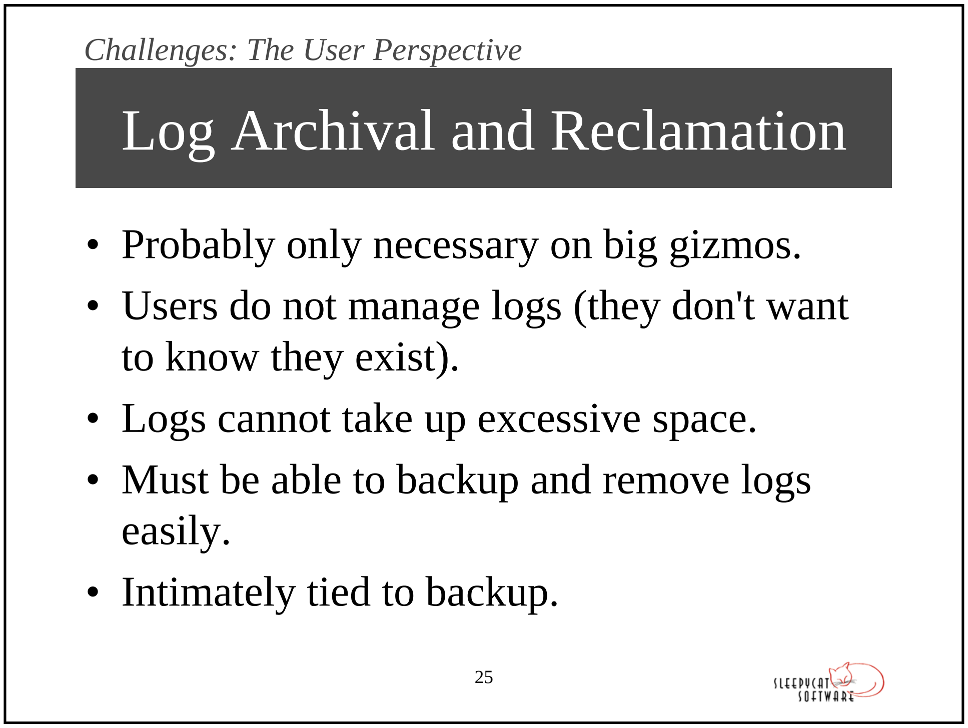
Backup on Big Gizmos

- Fairly traditional meaning
  - – Create a consistent snapshot of the database
  - – Snapshots taken hourly, daily, weekly, etc.
- Special requirements
  - – Hot backups
  - – Restoration on a different system

Challenges: The User Perspective

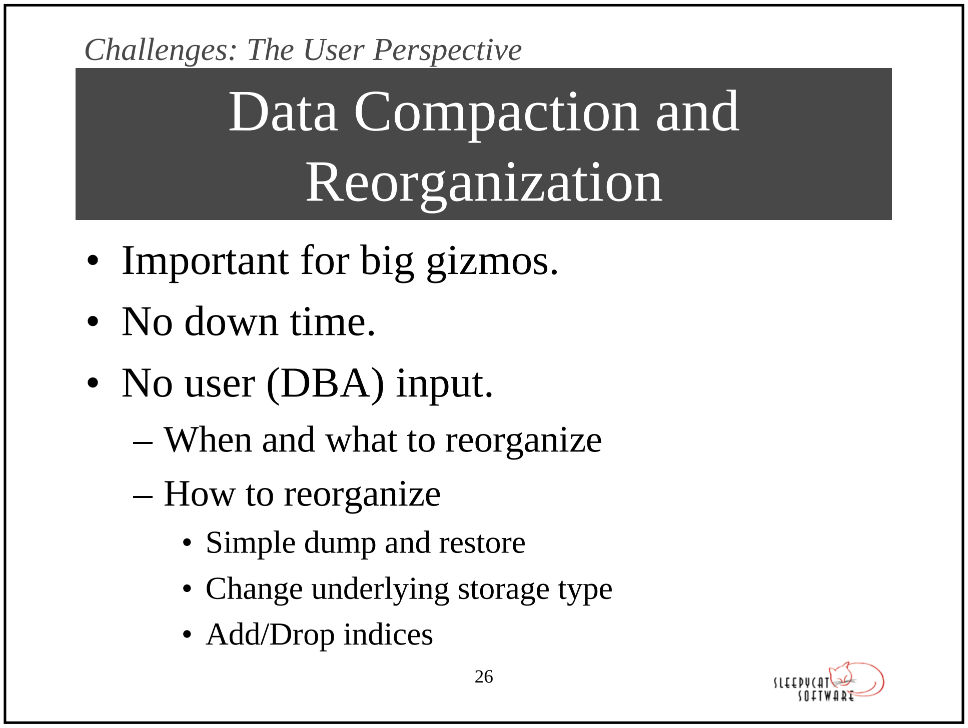
Backup on Small Gizmos

- This is not your standard tape backup!
- Opportunistic synchronization.
- Explicit synchronization.
- Backup to a remote repository.



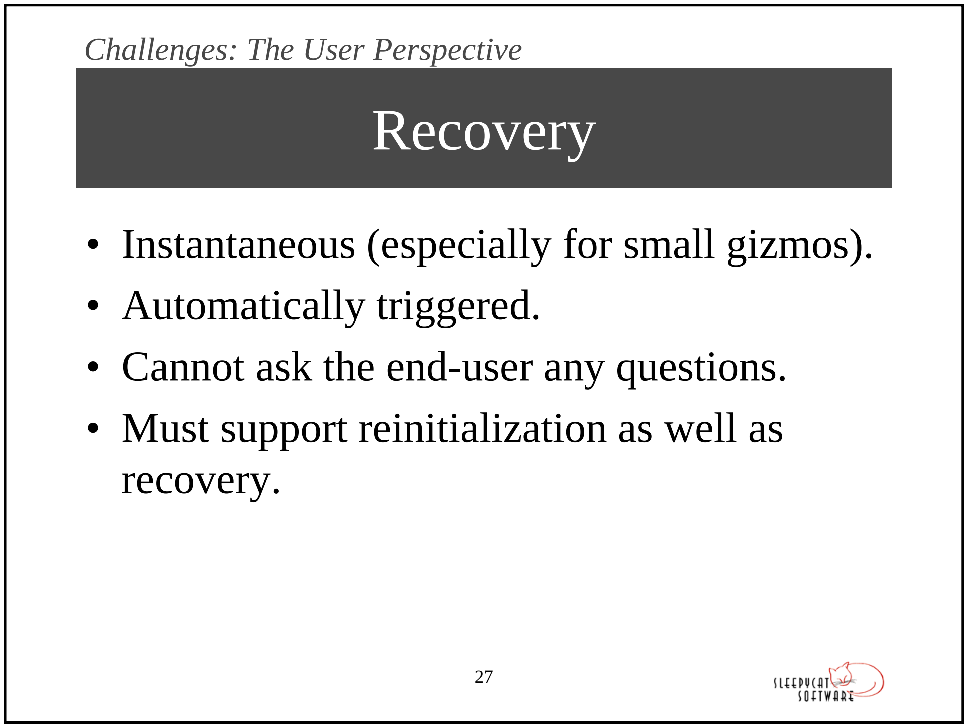
Log Archival and Reclamation

- Probably only necessary on big gizmos.
- Users do not manage logs (they don't want to know they exist).
- Logs cannot take up excessive space.
- Must be able to backup and remove logs easily.
- Intimately tied to backup.



Data Compaction and Reorganization

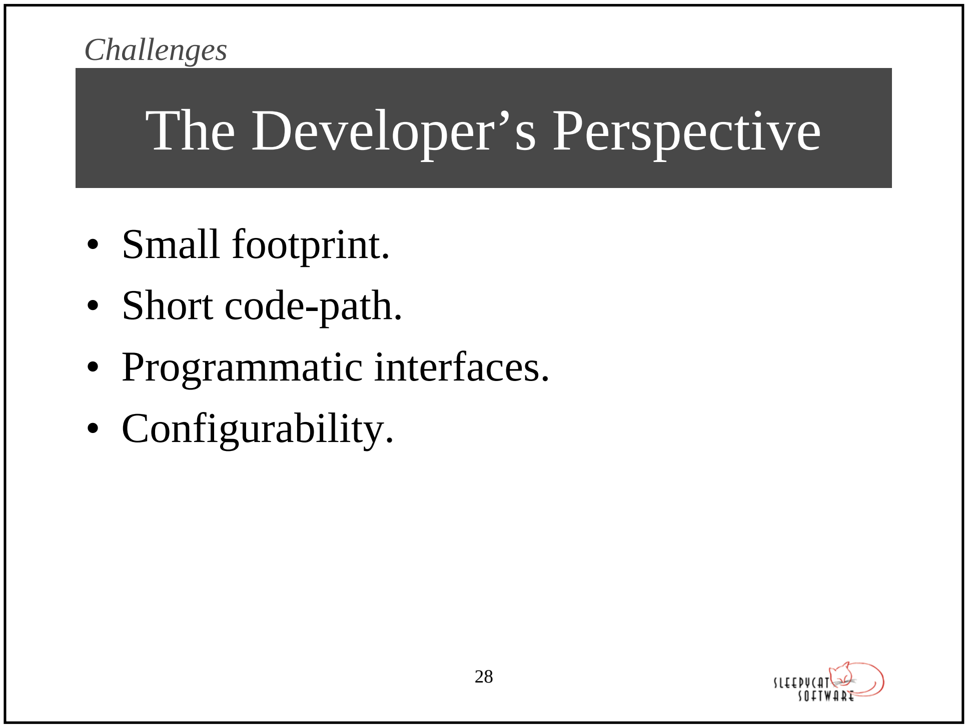
- Important for big gizmos.
- No down time.
- No user (DBA) input.
  - When and what to reorganize
  - How to reorganize
    - Simple dump and restore
    - Change underlying storage type
    - Add/Drop indices



Challenges: The User Perspective

Recovery

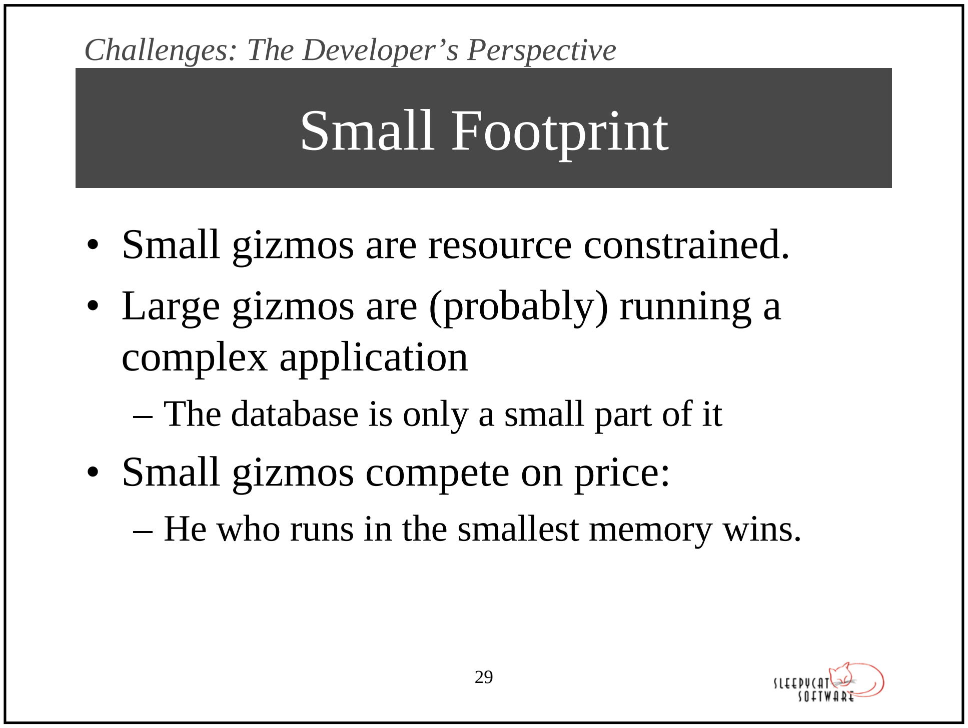
- Instantaneous (especially for small gizmos).
- Automatically triggered.
- Cannot ask the end-user any questions.
- Must support reinitialization as well as recovery.



Challenges

The Developer’s Perspective

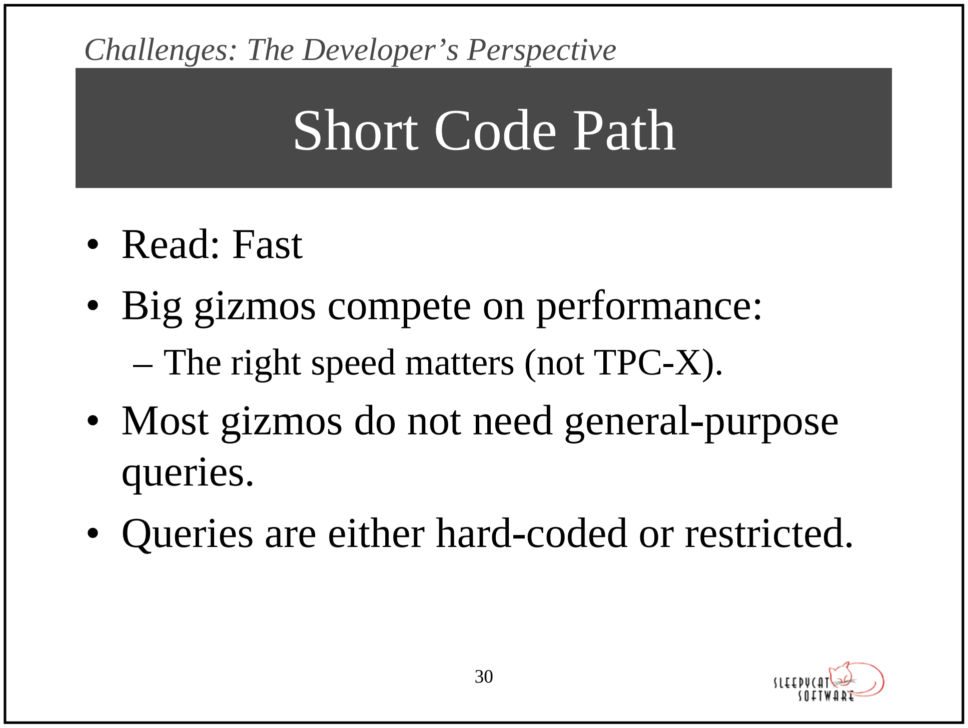
- Small footprint.
- Short code-path.
- Programmatic interfaces.
- Configurability.



Challenges: The Developer's Perspective

Small Footprint

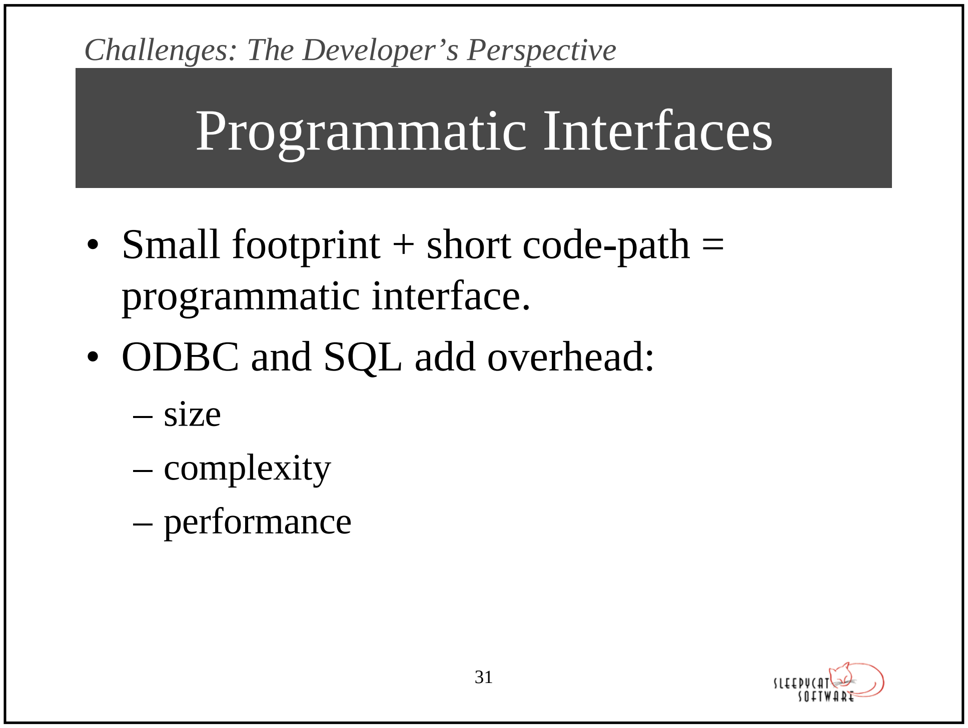
- Small gizmos are resource constrained.
- Large gizmos are (probably) running a complex application
  - The database is only a small part of it
- Small gizmos compete on price:
  - He who runs in the smallest memory wins.



Challenges: The Developer's Perspective

Short Code Path

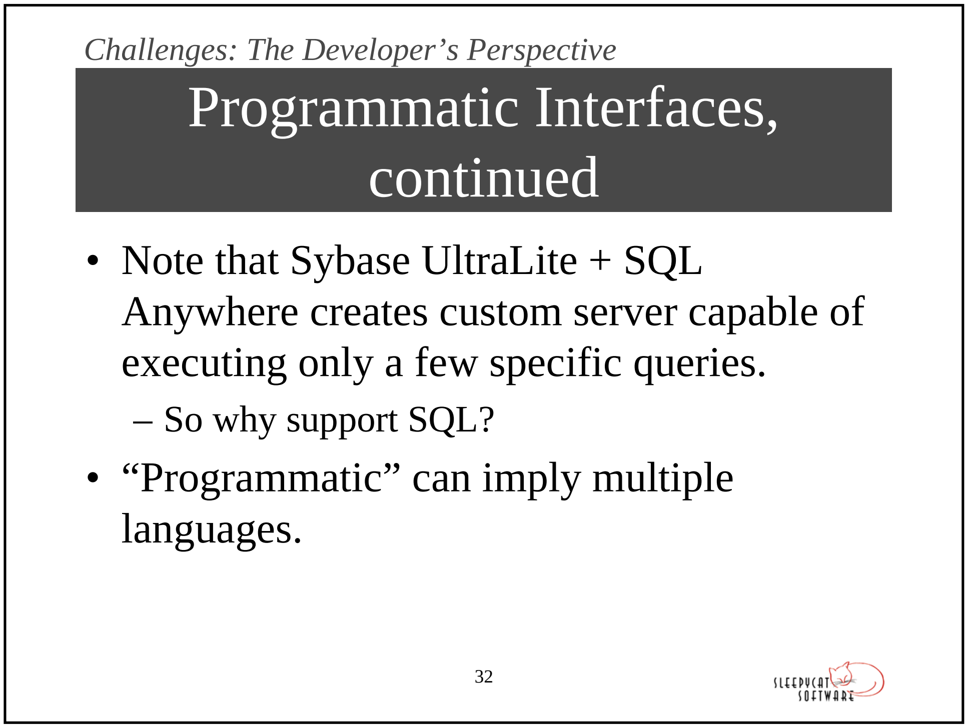
- Read: Fast
- Big gizmos compete on performance:
  - The right speed matters (not TPC-X).
- Most gizmos do not need general-purpose queries.
- Queries are either hard-coded or restricted.



Challenges: The Developer's Perspective

Programmatic Interfaces

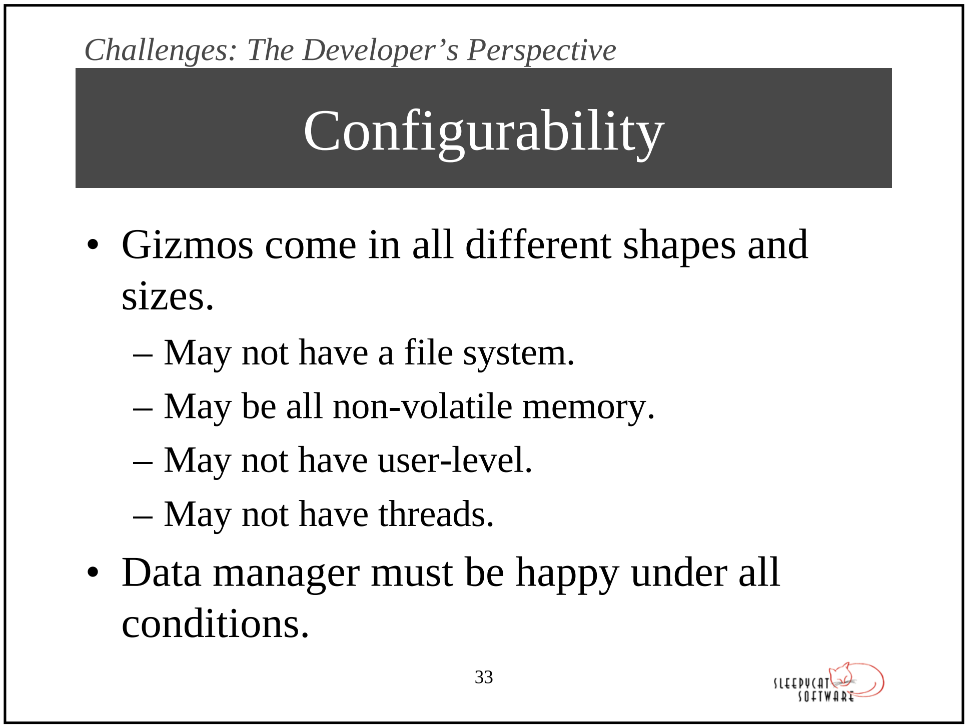
- Small footprint + short code-path = programmatic interface.
- ODBC and SQL add overhead:
  – size
  – complexity
  – performance



Challenges: The Developer's Perspective

Programmatic Interfaces, continued

- Note that Sybase UltraLite + SQL Anywhere creates custom server capable of executing only a few specific queries.
  – So why support SQL?
- "Programmatic" can imply multiple languages.

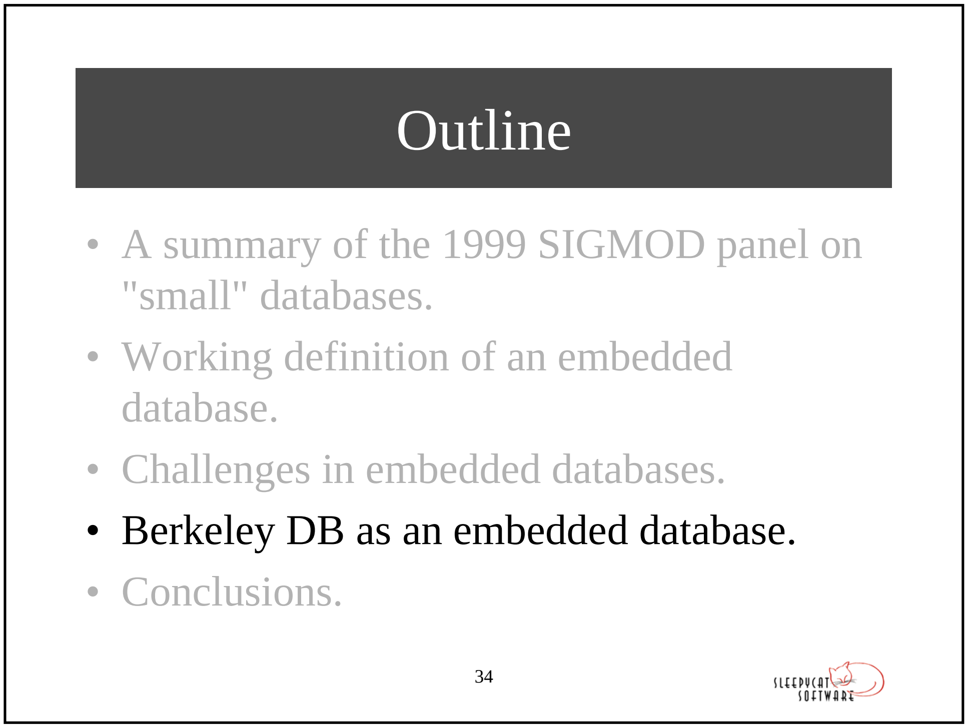


Challenges: The Developer’s Perspective

Configurability

- Gizmos come in all different shapes and sizes.
  - May not have a file system.
  - May be all non-volatile memory.
  - May not have user-level.
  - May not have threads.
- Data manager must be happy under all conditions.

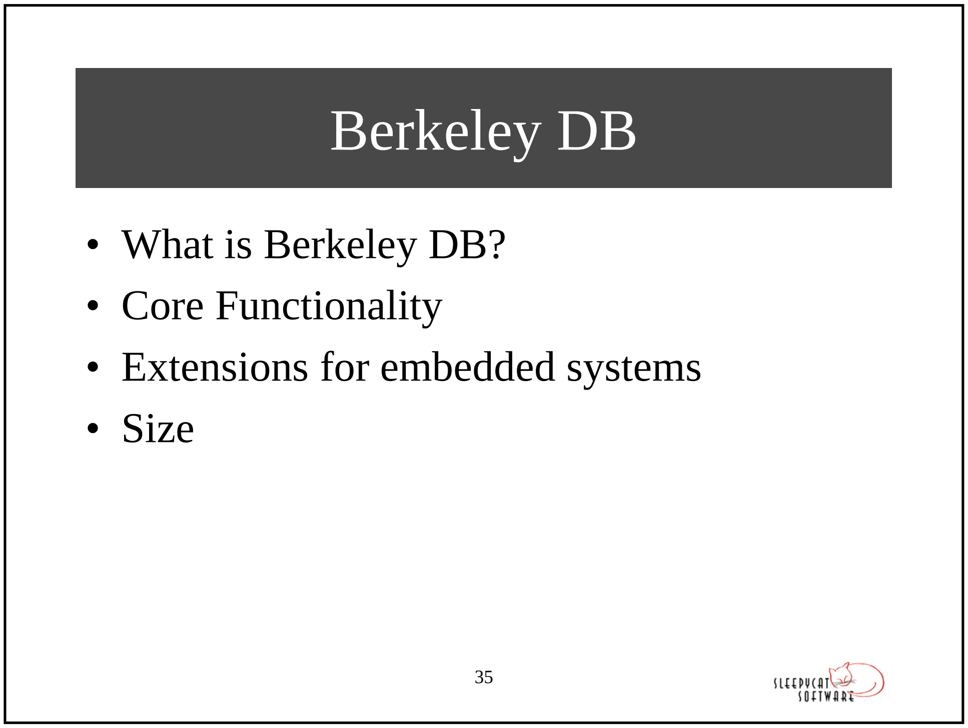
33



Outline

- A summary of the 1999 SIGMOD panel on "small" databases.
- Working definition of an embedded database.
- Challenges in embedded databases.
- Berkeley DB as an embedded database.
- Conclusions.

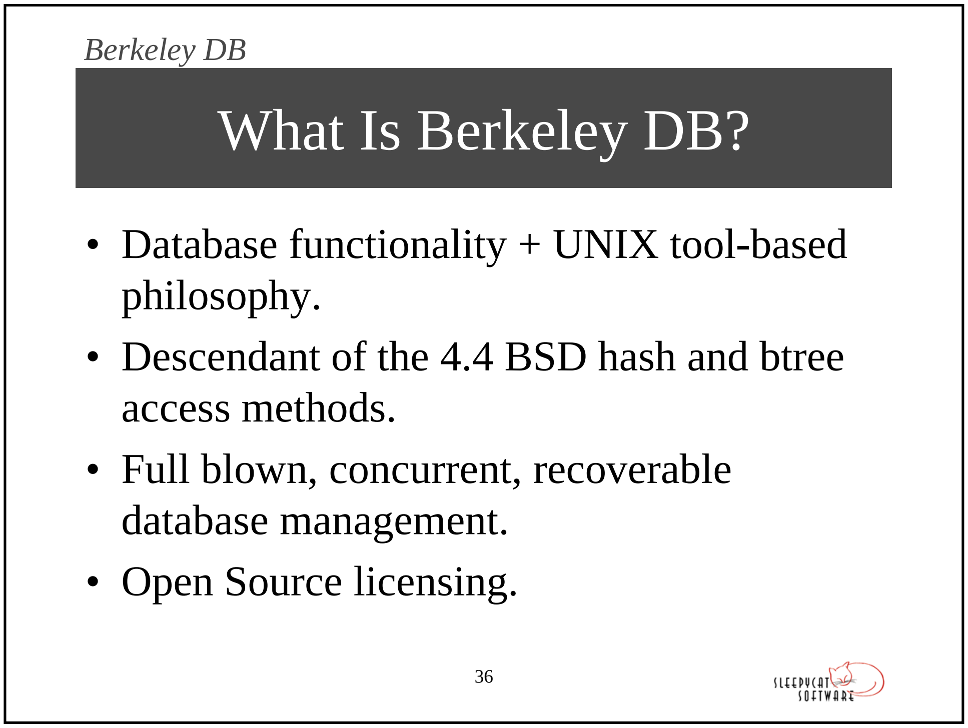
34



Berkeley DB

- What is Berkeley DB?
- Core Functionality
- Extensions for embedded systems
- Size

35

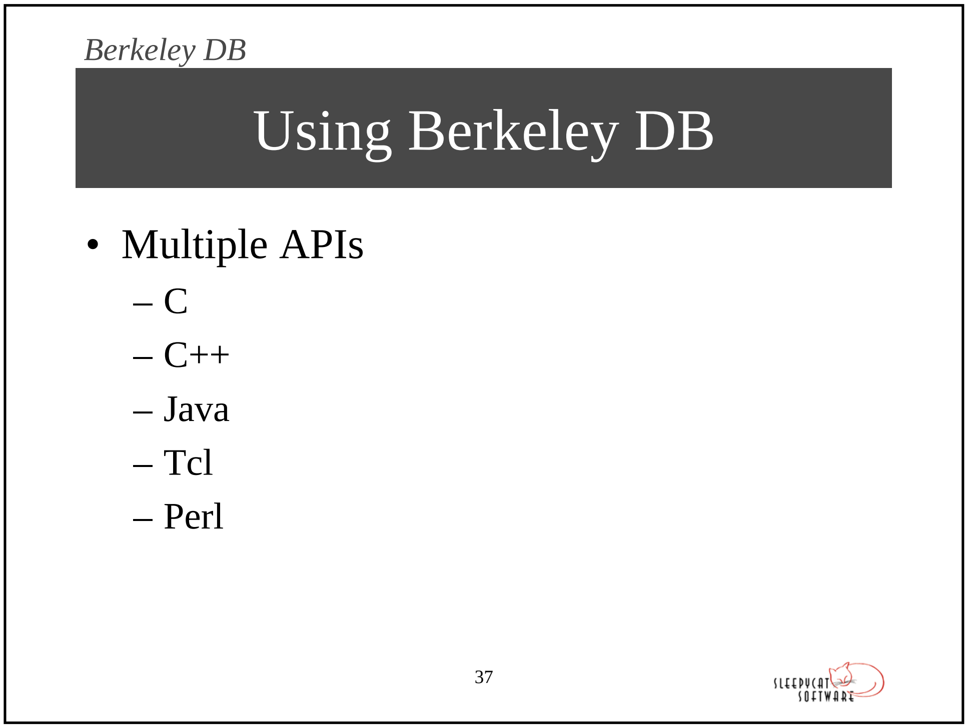


Berkeley DB

What Is Berkeley DB?

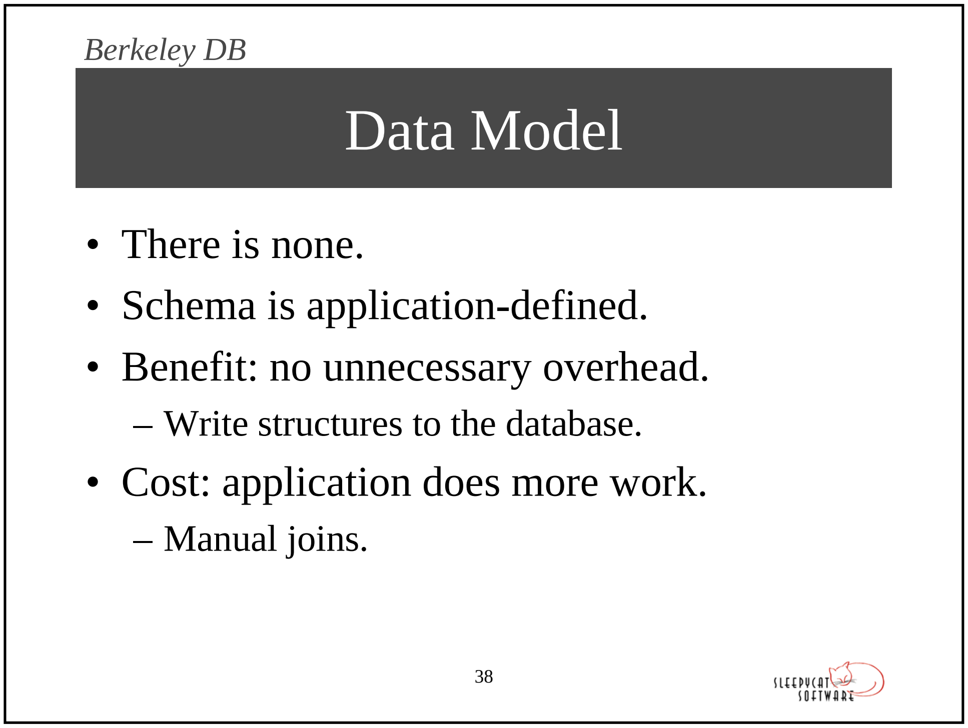
- Database functionality + UNIX tool-based philosophy.
- Descendant of the 4.4 BSD hash and btree access methods.
- Full blown, concurrent, recoverable database management.
- Open Source licensing.

36



Using Berkeley DB

- Multiple APIs
  - C
  - C++
  - Java
  - Tcl
  - Perl



Data Model

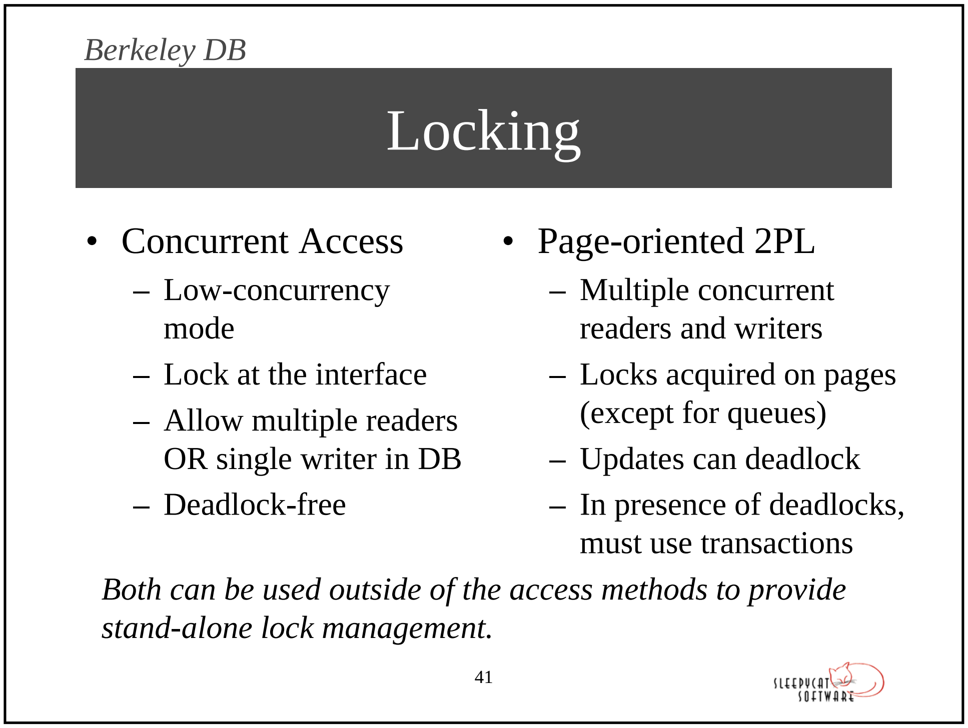
- There is none.
- Schema is application-defined.
- Benefit: no unnecessary overhead.
  - Write structures to the database.
- Cost: application does more work.
  - Manual joins.

Core Functionality

- Access methods
- Locking
- Logging
- Shared buffer management
- Transactions
- Utilities

Access Methods

- B+ Trees: in-order optimizations.
- Dynamic Linear Hashing.
- Fixed & Variable Length Records.
- High concurrency queues.

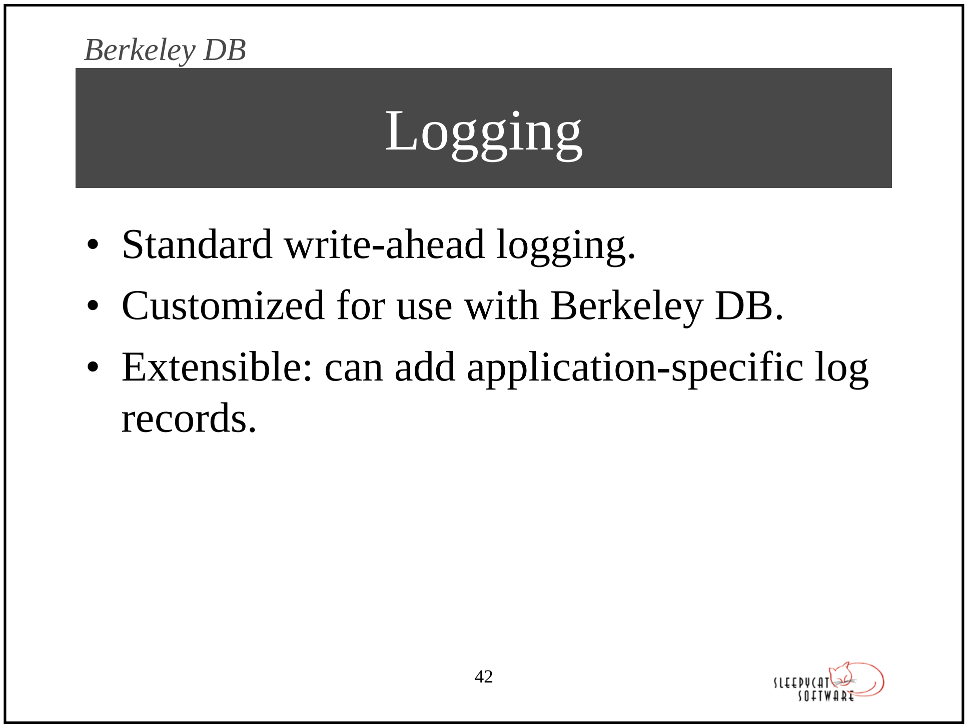


Berkeley DB

Locking

- Concurrent Access
  - Low-concurrency mode
  - Lock at the interface
  - Allow multiple readers OR single writer in DB
  - Deadlock-free
- Page-oriented 2PL
  - Multiple concurrent readers and writers
  - Locks acquired on pages (except for queues)
  - Updates can deadlock
  - In presence of deadlocks, must use transactions

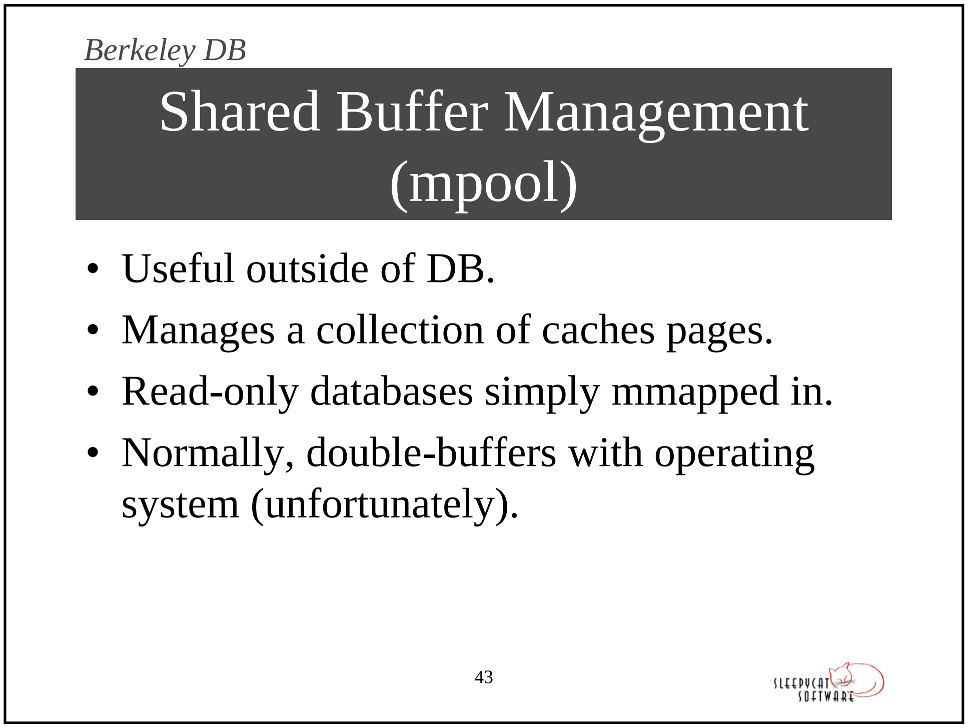
Both can be used outside of the access methods to provide stand-alone lock management.



Berkeley DB

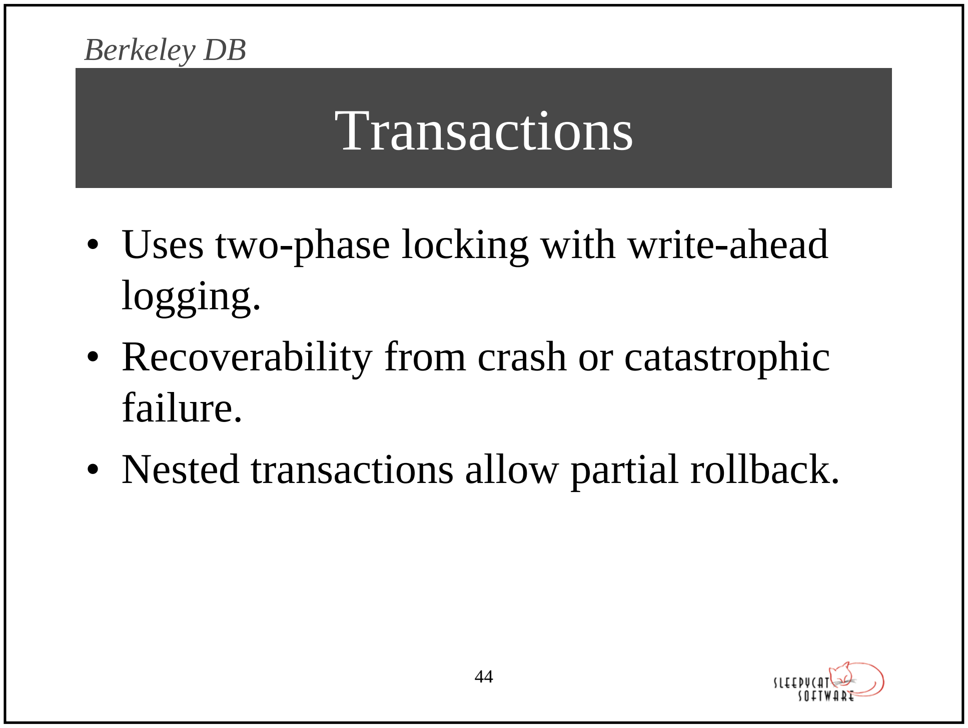
Logging

- Standard write-ahead logging.
- Customized for use with Berkeley DB.
- Extensible: can add application-specific log records.



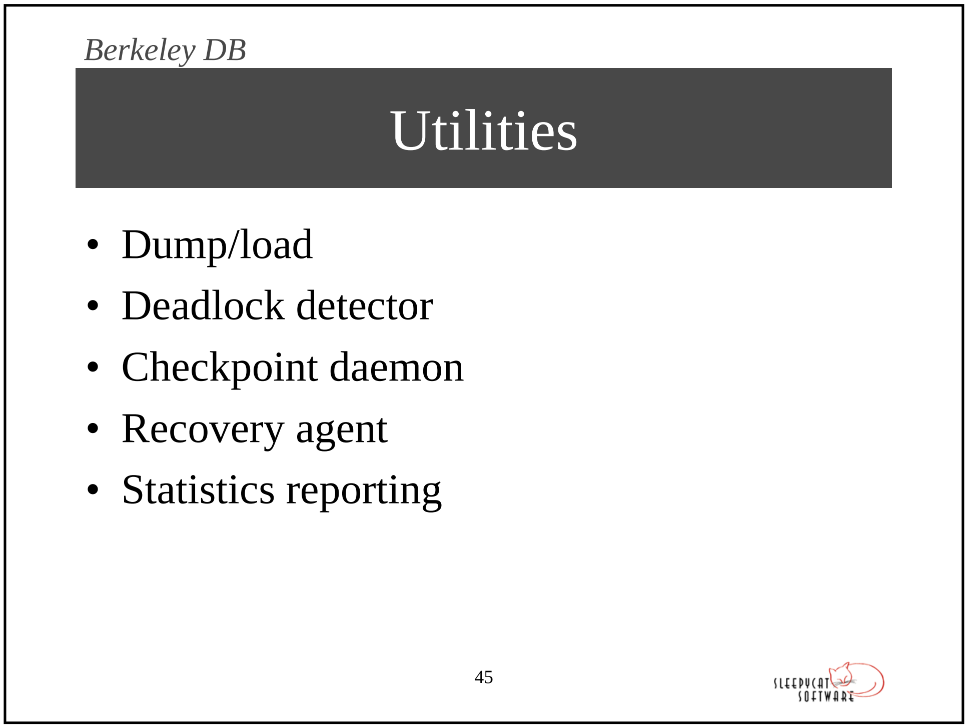
Shared Buffer Management (mpool)

- Useful outside of DB.
- Manages a collection of caches pages.
- Read-only databases simply mmapped in.
- Normally, double-buffers with operating system (unfortunately).



Transactions

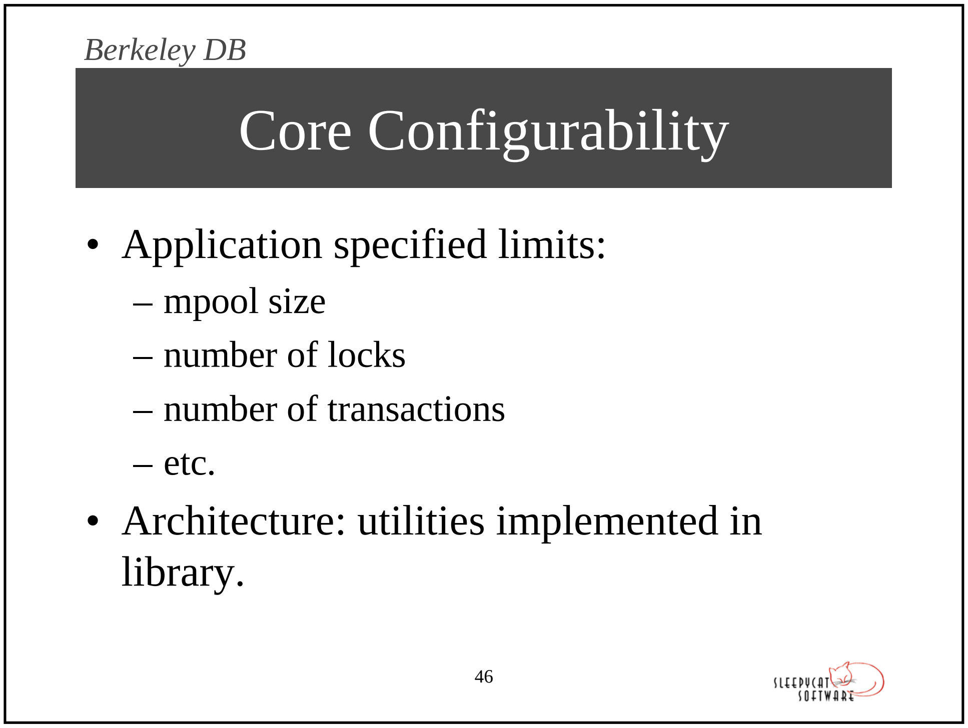
- Uses two-phase locking with write-ahead logging.
- Recoverability from crash or catastrophic failure.
- Nested transactions allow partial rollback.



Berkeley DB

Utilities

- Dump/load
- Deadlock detector
- Checkpoint daemon
- Recovery agent
- Statistics reporting



Berkeley DB

Core Configurability

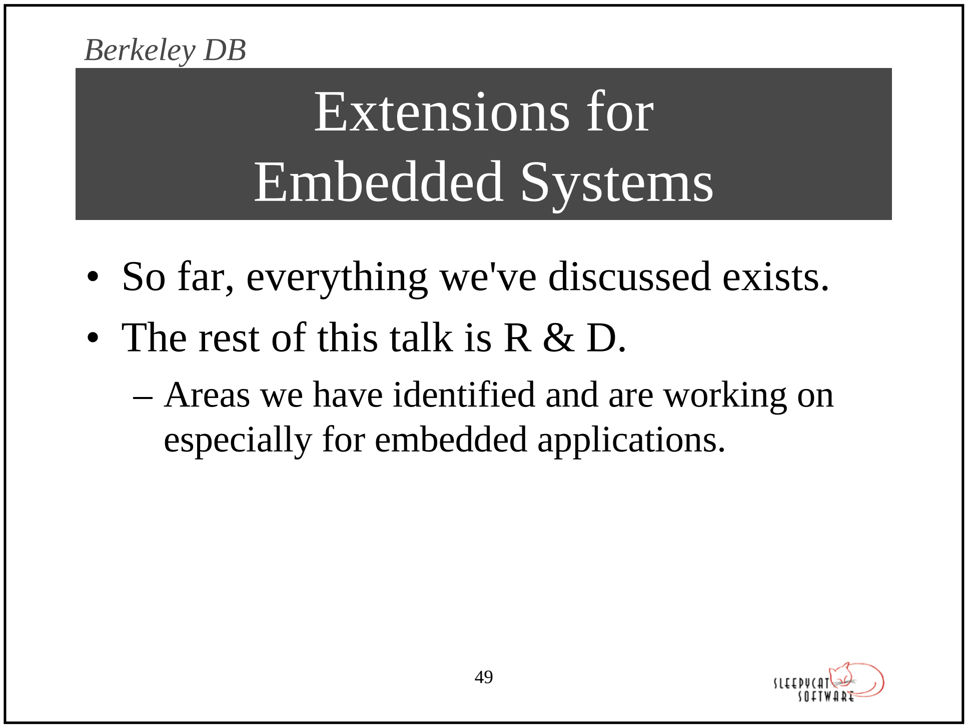
- Application specified limits:
  - mpool size
  - number of locks
  - number of transactions
  - etc.
- Architecture: utilities implemented in library.

Configuring the Access Methods

- Btrees:
  - sort order: application-specified functions.
  - compression: application-specified functions.
- Hash:
  - application-specified hash functions.
  - pre-allocate buckets if size is specified.

Configuring OS Interaction

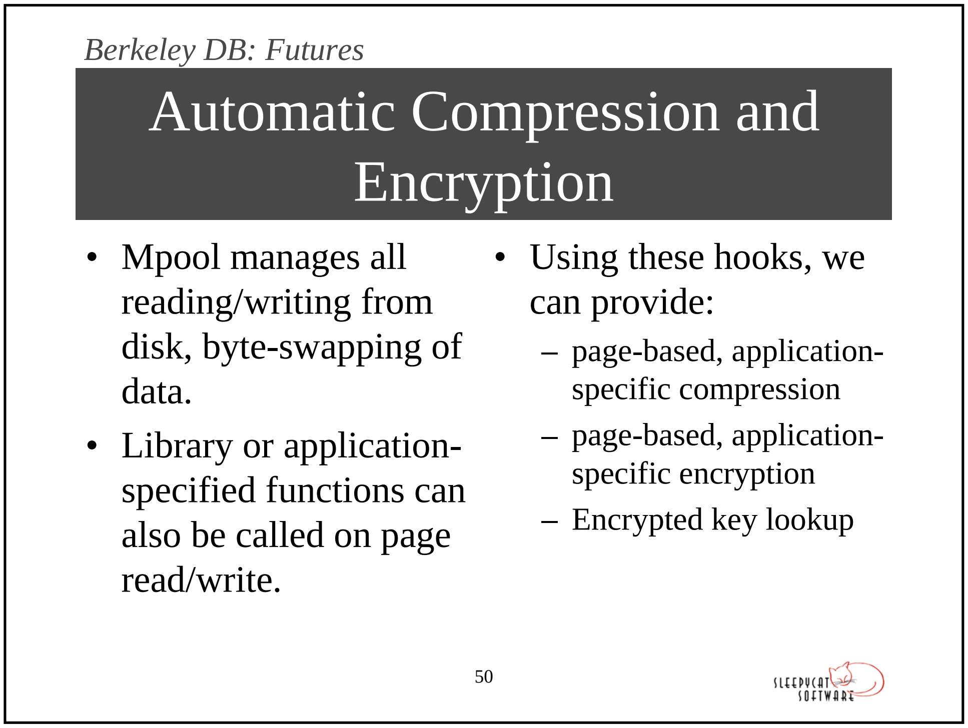
- File system
  - explicitly locate log files
  - explicitly locate data files
  - control over page sizes
  - etc.
- Shared memory
  - specify shared memory architecture (mmap, shmget, malloc).



Berkeley DB

Extensions for Embedded Systems

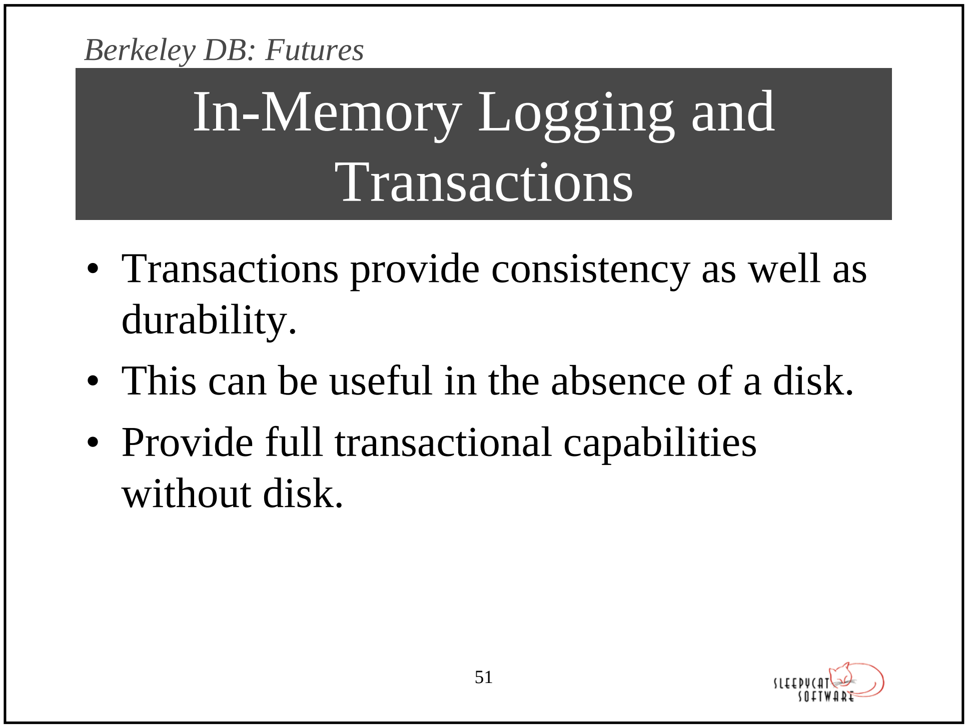
- So far, everything we've discussed exists.
- The rest of this talk is R & D.
  - Areas we have identified and are working on especially for embedded applications.



Berkeley DB: Futures

Automatic Compression and Encryption

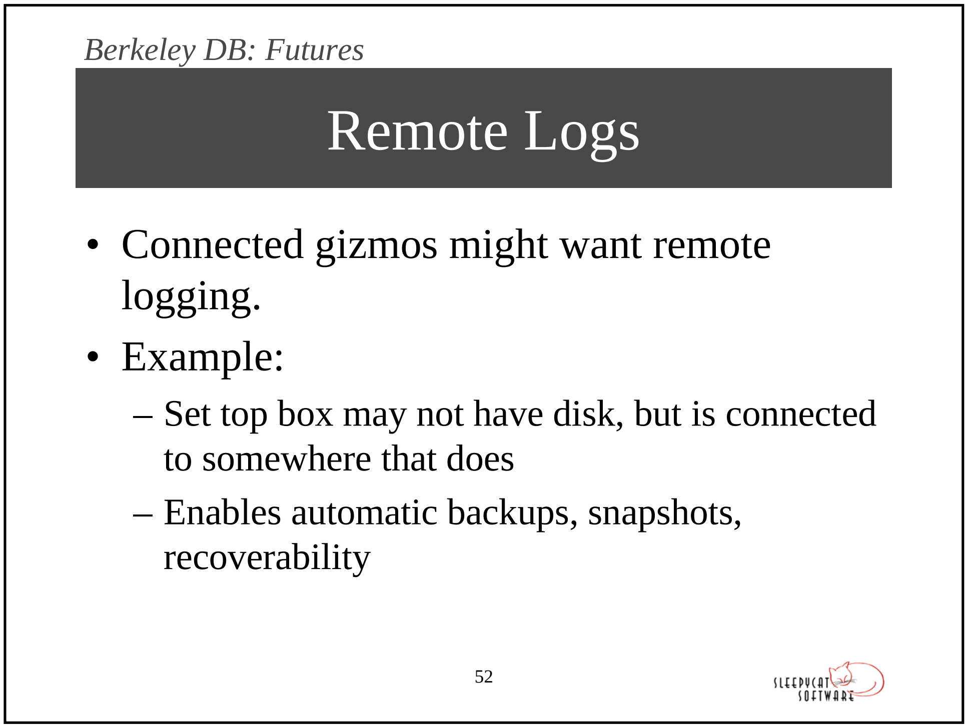
- Mpool manages all reading/writing from disk, byte-swapping of data.
- Library or application-specified functions can also be called on page read/write.
- Using these hooks, we can provide:
  - page-based, application-specific compression
  - page-based, application-specific encryption
  - Encrypted key lookup



Berkeley DB: Futures

In-Memory Logging and Transactions

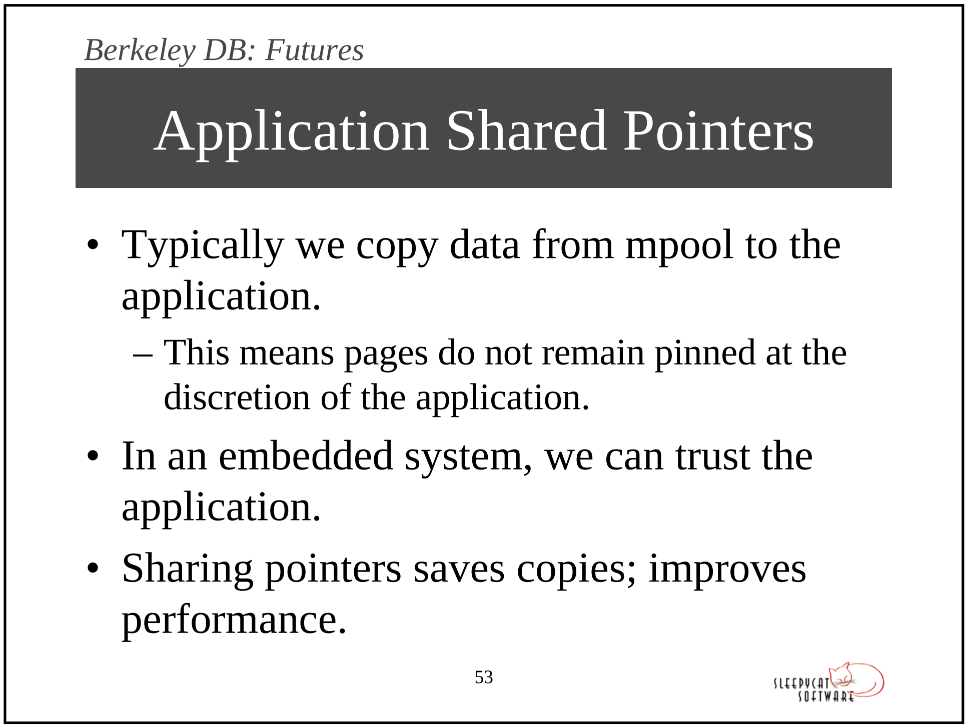
- Transactions provide consistency as well as durability.
- This can be useful in the absence of a disk.
- Provide full transactional capabilities without disk.



Berkeley DB: Futures

Remote Logs

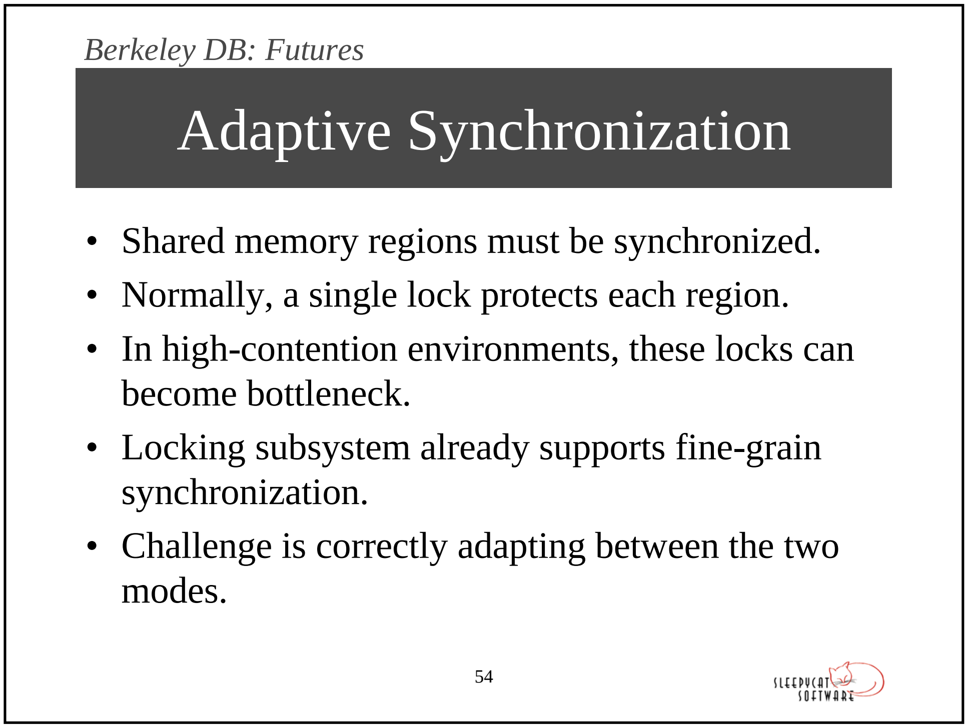
- Connected gizmos might want remote logging.
- Example:
  - Set top box may not have disk, but is connected to somewhere that does
  - Enables automatic backups, snapshots, recoverability



Berkeley DB: Futures

Application Shared Pointers

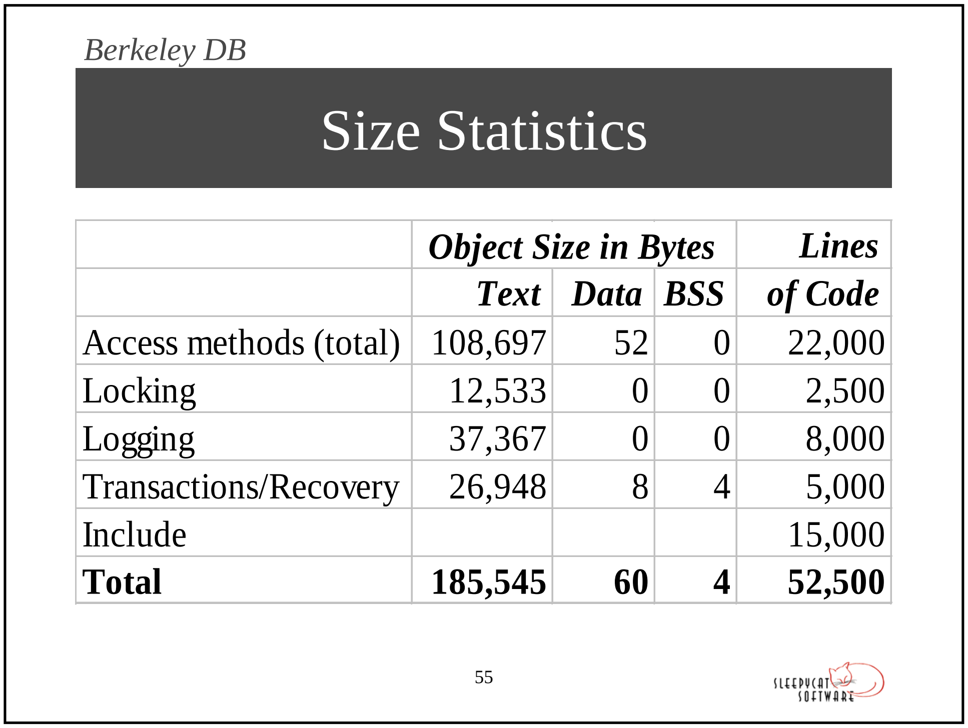
- Typically we copy data from mpool to the application.
  - This means pages do not remain pinned at the discretion of the application.
- In an embedded system, we can trust the application.
- Sharing pointers saves copies; improves performance.



Berkeley DB: Futures

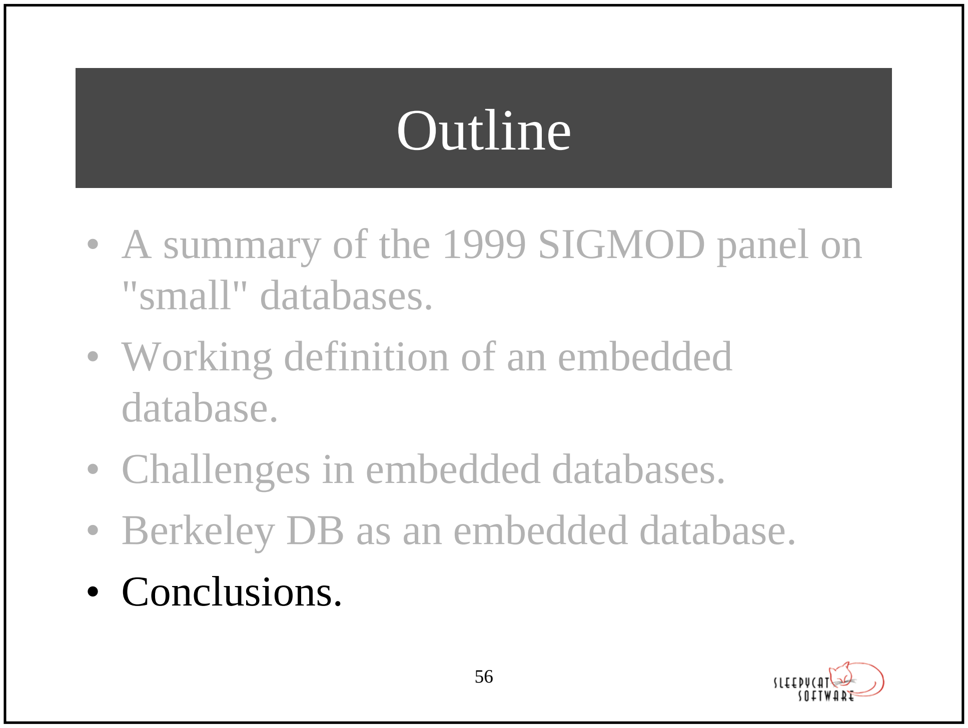
Adaptive Synchronization

- Shared memory regions must be synchronized.
- Normally, a single lock protects each region.
- In high-contention environments, these locks can become bottleneck.
- Locking subsystem already supports fine-grain synchronization.
- Challenge is correctly adapting between the two modes.



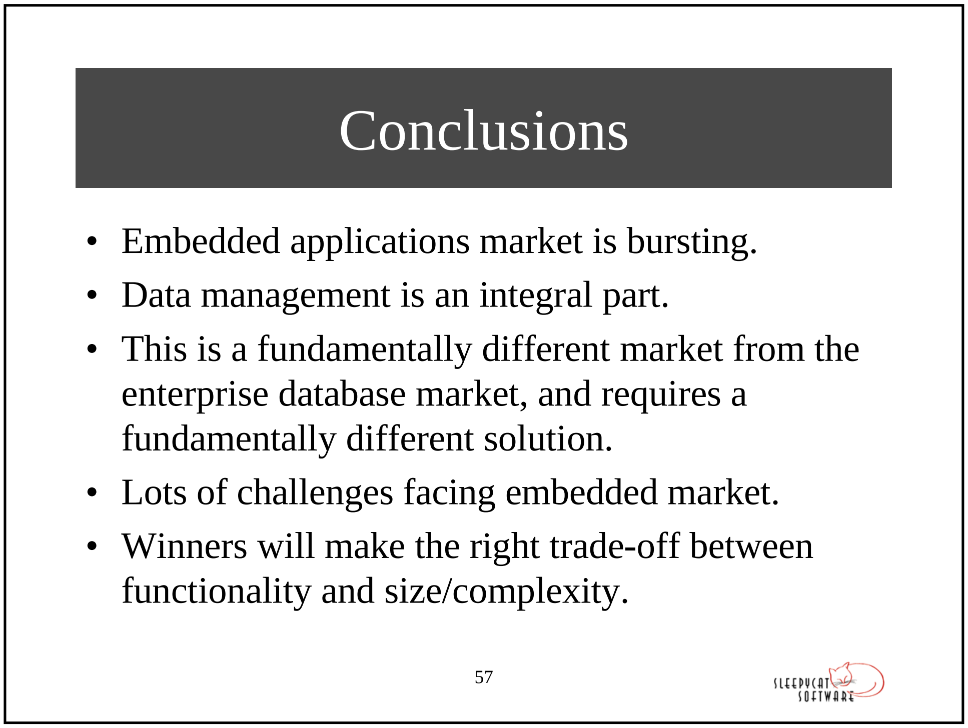
Size Statistics

	Object Size in Bytes			*Lines*
	Text	*Data*	*BSS*	*of Code*
Access methods (total)	108,697	52	0	22,000
Locking	12,533	0	0	2,500
Logging	37,367	0	0	8,000
Transactions/Recovery	26,948	8	4	5,000
Include				15,000
Total	**185,545**	**60**	**4**	**52,500**



Outline

- A summary of the 1999 SIGMOD panel on "small" databases.
- Working definition of an embedded database.
- Challenges in embedded databases.
- Berkeley DB as an embedded database.
- Conclusions.



Conclusions

- Embedded applications market is bursting.
- Data management is an integral part.
- This is a fundamentally different market from the enterprise database market, and requires a fundamentally different solution.
- Lots of challenges facing embedded market.
- Winners will make the right trade-off between functionality and size/complexity.

SLEEPYCAT SOFTWARE

…the embedded database company tm

Come visit us in Booth #401!

Margo Seltzer
Sleepycat Software
margo@sleepycat.com
http://www.sleepycat.com